```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPrPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
```

```
**FILE**ID**UETDR7800                                   H 13                                                              UE
                                                                                                                          VO

                                                                                                                          49

UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    77777777      888888      000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    77777777      888888      000000      000000
UU      UU  EE              TT      DD      DD  RR      RR          77  88      88  00      00  00      00
UU      UU  EE              TT      DD      DD  RR      RR          77  88      88  00      00  00      00
UU      UU  EE              TT      DD      DD  RR      RR          77  88      88  00    0000  00    0000
UU      UU  EE              TT      DD      DD  RR      RR          77  88      88  00    0000  00    0000
UU      UU  EEEEEEEE        TT      DD      DD  RRRRRRRR          77      888888      00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      DD      DD  RRRRRRRR          77      888888      00  00  00  00  00  00
UU      UU  EE              TT      DD      DD  RR  RR          77      88      88  0000    00  0000    00
UU      UU  EE              TT      DD      DD  RR  RR          77      88      88  0000    00  0000    00
UU      UU  EE              TT      DD      DD  RR    RR      77        88      88  00      00  00      00      ....      4E
UU      UU  EE              TT      DD      DD  RR    RR      77        88      88  00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    RR      RR  77            888888      000000      000000        ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    RR      RR  77            888888      000000      000000        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS                                                                                                4E
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS                                                                                          4E
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE UETDR7800 VAX/VMS UETP DEVICE TEST FOR DR780/DR750
0000            .IDENT  'V04-000'
0000            .ENABLE SUPPRESSION
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:
0000    ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    ;       account.
0000    ;
0000    ; ABSTRACT:
0000    ;       This module exercises a DR780 or a DR750 in the VAX/VMS system using
0000    ;       QIO functions.  2048(10) byte transfers are written and read
0000    ;       using chained command packets.  The transfers are verified for
0000    ;       correct data.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       This program will run in user access mode, with interrupts enabled
0000    ;       at all times.  This program  must be linked with SYS$SYSTEM:SYS.STB
0000    ;       because of its use of symbol IOC$GW_XFMXRATE to get the current DR
0000    ;       max transfer rate. This program requires the following privileges and
0000    ;       quotas:
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR: Larry Jones,  CREATION DATE: May, 1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-010 RNH0008         Richard N. Holstein,    21-Mar-1984
0000    ;               Change wording on error messages.
0000    ;
0000    ;       V03-009 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    ;               Take advantage of the new UETP message codes.  Fix SSERROR
```

```
0000  ;                 interaction with RMS_ERROR.
0000  ;
0000  ;          V03-008 RNH0006         Richard N. Holstein,    05-Jan-1984
0000  ;                 Set up SYS$ERROR for the ucode loader process and report back
0000  ;                 any results.
0000  ;
0000  ;          V03-007 RNH0005         Richard N. Holstein,    19-Dec-1983
0000  ;                 Give correct sentinels to Test Controller.
0000  ;
0000  ;          V03-006 RNH0004         Richard N. Holstein,    21-Nov-1983
0000  ;                 Use decimal conversion routine for unit numbers.
0000  ;
0000  ;          V03-005 RNH0003         Richard N. Holstein,    11-Mar-1983
0000  ;                 Don't signal ending message in EXIT_HANDLER.
0000  ;
0000  ;          V03-004 RNH0002         Richard N. Holstein,    25-Feb-1983
0000  ;                 Allow for longer device names.
0000  ;
0000  ;          V03-003 RNH0001         Richard N. Holstein,    15-Oct-1982
0000  ;                 Miscellaneous fixes listed in the V3B UETP Workplan.
0000  ;
0000  ;          V03-002 LDJ0002         Larry D. Jones,         11-Mar-1982
0000  ;                 Fixed missing bit set in command table for DR.
0000  ;
0000  ;          V03-001 LDJ0001         Larry D. Jones,         29-Sep-1981
0000  ;                 Filled in error path exits with missing STATUS values and
0000  ;                 reversed the order of the error/end test.
0000  ;
0000  ;**
```

```
0000            .SBTTL  Declarations
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000    ;       SYS$LIBRARY:LIB.MLB      for general definitions
0000    ;       SHRLIB$:UETP.MLB         for UETP definitions
0000
0000    ;
0000    ; MACROS:
0000    ;
0000            $ACCDEF                             ; Accounting definitions
0000            $CHFDEF                             ; Condition handler frame definitions
0000            $DEVDEF                             ; Device definitions
0000            $DIBDEF                             ; Device Information Block
0000            $DVIDEF                             ; $GETDVI ITMLST item codes
0000            $SHRDEF                             ; Shared messages
0000            $SSDEF                              ; System Service status codes
0000            $STSDEF                             ; Status return
0000            $UETUNTDEF                          ; UETP unit block offset definitions
0000            $UETPDEF                            ; UETP
0000            $XFDEF                              ; DR780 definitions
0000    ;
0000    ; USER MACRO DEFINITIONS
0000    ;
0000    ; QRETRY - This macro executes an interlocked queue instruction and
0000    ;            retries up to 25 times if the queue is locked.
0000    ; INPUTS:
0000    ;       OPCODE = OPCODE NAME : INSQHI, INSQTI, REMQHI, REMQTI.
0000    ;       OPERAND1 = first operand for opcode.
0000    ;       OPERAND2 = second operand for opcode.
0000    ;       SUCCESS = label to branch to if operation succeeds (may be defaulted).
0000    ;       ERROR  = label to branch to if operation fails (may be omitted).
0000    ;
0000    ; OUTPUTS:
0000    ;       R0     = destroyed.
0000    ;       C-BIT = clear if operation succeeded.
0000    ;               set if operation failed - queue locked.
0000    ;               (must be checked before V-bit or Z-bit)
0000    ;
0000    ;       REMQTI OR REMQHI:
0000    ;
0000    ;       V-bit = clear if an entry removed from queue.
0000    ;               set if no entry removed from the queue.
0000    ;
0000    ;       INSQTI OR INSQHI:
0000    ;
0000    ;       Z-bit = clear if entry is not first in the queue.
0000    ;               set if entry is first in the queue.
0000    ;
0000                    .MACRO  QRETRY OPCODE,OPERAND1,OPERAND2,SUCCESS,ERROR,?LOOP,?OK
0000                    CLRL    R0
0000   LOOP:
0000                    OPCODE  OPERAND1,OPERAND2
0000                    .IF     NB      SUCCESS
0000                    BCC     SUCCESS
0000                    .IFF
0000                    BCC     OK
```

```
                0000                      .ENDC
                0000                      AOBLSS  #25,R0,LOOP
                0000                      .IF     NB      ERROR
                0000                      BRW     ERROR
                0000                      .ENDC
                0000      OK:
                0000              .ENDM   QRETRY
                0000      ;
                0000              .MACRO BUILD,NAME,STATUS
                0000                      .=PC1...                ; PC of ASCIC pkt NAME table
                0000                      .ADDRESS PC2...         ; ASCIC pkt address
                0000                      .LONG   ^X'STATUS       ; Expected packet return code
                0000                      PC1...=PC1...+8         ; Bump to the next address
                0000                      .=PC2...                ; Point to the next ASCIC msg
                0000                                              ; Make it's label and function ID
                0000      NAME:
                0000                      .ASCIC /NAME, /
                0000                      PC2...=.                ; Update the string PC
                0000              .ENDM BUILD
                0000      ;
                0000      ; EQUATED SYMBOLS:
                0000      ;
                0000      ;   Facility number definitions:
      00000001  0000              RMS$_FACILITY = 1
                0000
                0000      ;   SHR message definitions:
      00740000  0000              UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
      007410E0  0000              UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
      00741038  0000              UETP$_BEGIND = UETP!SHR$_BEGIND
      00741080  0000              UETP$_ENDEDD = UETP!SHR$_ENDEDD
      00741098  0000              UETP$_OPENIN = UETP!SHR$_OPENIN
      00741130  0000              UETP$_TEXT   = UETP!SHR$_TEXT
                0000
                0000      ;   Internal flag bits...:
      00000000  0000              CC_FLGV       = 0               ; Set when a control C is typed
      00000001  0000              TEST_OVERV    = 1               ; Set when test is over
      00000002  0000              SAFE_TO_UPDV  = 2               ; Set if it's safe to update UETINIDEV
      00000003  0000              ERR_FLGV      = 3               ; Set when an error occurs
      00000004  0000              FPAC_FLGV     = 4               ; Set when first packet is serviced
      00000005  0000              BEGIN_MSGV    = 5               ; Set if "BEGIN" msg has been printed
                0000
                0000      ;   ...and corresponding masks:
      00000001  0000              CC_FLGM       = 1@CC_FLGV
      00000002  0000              TEST_OVERM    = 1@TEST_OVERV
      00000004  0000              SAFE_TO_UPDM  = 1@SAFE_TO_UPDV
      00000008  0000              ERR_FLGM      = 1@ERR_FLGV
      00000010  0000              FPAC_FLGM     = 1@FPAC_FLGV
      00000020  0000              BEGIN_MSGM    = 1@BEGIN_MSGV
                0000
                0000      ;   Miscellany:
      00000020  0000              LC_BITM         = ^X20          ; Mask to convert lower case to upper
      0000001B  0000              ESC             = ^X1B          ; Escape definition
      00000028  0000              REC_SIZE        = 40            ; UETINIDEV.DAT record size
      00000084  0000              TEXT_BUFFER     = 132           ; Internal text buffer size
      00000003  0000              SS_SYNCH_EFN    = 3             ; Synch miscellaneous system services
      0000000F  0000              MAX_PROC_NAME   = 15            ; Longest process name
      0000000A  0000              MAX_DEV_DESIG   = 10            ; Longest possible controller name
```

UE
V0

```
00000005  0000   202         MAX_UNIT_DESIG= 5                 ; Longest possible unit number
00000011  0000   203         NO_OF_POS_PKTS= 17                ; Number of possible packet types
0000000D  0000   204         PKT_COUNT     = 13                ; Number of packets to be processed
00006030  0000   205         UNUSED_FUNC   = ^X6030            ; Bit mask of the unused functions
00000800  0000   206         BUFSIZ        = 2048              ; Read/write buffer size
00000001  0000   207         EFN1          = 1                 ; EF number definitions
00000004  0000   208         EFN2          = 4                 ; EFN used for three minute timer
          0000   209
          0000   210 ;   The following definitions are set depending on the device under test.
          0000   211
00000000  0000   212         DEVDEP_SIZE   = 0                 ; Size of device dependent part of UETUNT
00000800  0000   213         WRITE_SIZE    = BUFSIZ            ; Size of device write buffer
00000800  0000   214         READ_SIZE     = BUFSIZ            ; Size of device read buffer
          0000   215
          0000   216         PAGES = <<UETUNT$C_INDSIZ+-       ; Add together all of the pieces...
          0000   217                   DEVDEP_SIZE+-           ; ...which make up a UETP unit block...
          0000   218                   WRITE_SIZE+-            ; ...to give to the $EXPREG service below
          0000   219                   READ_SIZE+-
00000009  0000   220                   511>/512>
          0000   221
```

```
                                     0000     223              .SBTTL  Read-Only Data
                                 00000000     224              .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                     0000     225
                                     0000     226 ACNT_NAME:                                      ; Process name on exit
53 45 54 53 59 53 00000008'010E0000' 0000     227              .ASCID  /SYSTEST/
                                  54 000E
                                     000F     228
                                     000F     229 TEST_NAME:                                      ; This test name
37 52 44 54 45 55 00000017'010E0000' 000F     230              .ASCID  /UETDR7800/
                            30 30 38 001D
                                     0020     231
                                     0020     232 SUPDEV_GBLSEC:                                  ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000' 0020     233              .ASCID  /UETSUPDEV/
                            56 45 44 002E
                                     0031     234
                                     0031     235 CONTROLLER:                                     ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000' 0031     236              .ASCID  /CTRLNAME/
                               45 4D 003F
                                     0041     237
                                     0041     238 PROCESS:                                        ; ucode load program
59 53 24 53 59 53 00000049'010E0000' 0041     239              .ASCID  /SYS$SYSTEM:XFLOADER.EXE/
45 44 41 4F 4C 46 58 3A 4D 45 54 53  004F
                   45 58 45 2E 52    005B
                                     0060     240
                                     0060     241 XFLDR_SYS$ERROR:                                ; File name of SYS$ERROR for XFLOADER
2E 52 4F 52 52 45 5F 52 44 4C 46 58  0060     242              .ASCII  /XFLDR_ERROR.LOG/
                            47 4F 4C 006C
                            0000000F 006F     243 XFLDR_SYS$ERROR_LENGTH = .-XFLDR_SYS$ERROR
                                     006F     244
                                     006F     245 XFLDR_SYS$ERROR_DESC:                           ; SYS$ERROR descriptor during $CREPRC
                           0000 000F 006F     246              .WORD   XFLDR_SYS$ERROR_LENGTH,0
                           00000060' 0073     247              .ADDRESS XFLDR_SYS$ERROR
                                     0077     248
                                     0077     249 XFLDR_HUNG:                                     ; We timed out waiting to load ucode
6D 20 32 33 52 44 0000007F'010E0000' 0077     250              .ASCID  /DR32 microcode loader process seems to be hung. /
61 6F 6C 20 65 64 6F 63 6F 72 63 69  0085
20 73 73 65 63 6F 72 70 20 72 65 64  0091
20 65 62 20 6F 74 20 73 6D 65 65 73  009D
                   20 2E 67 6E 75 68 00A9
                                     00AF     251
                                     00AF     252 XFLDR_LOG:                                      ; Error messages during ucode loading
6D 20 32 33 52 44 000000B7'010E0000' 00AF     253              .ASCID  /DR32 microcode loader process logged some error message(s): /
61 6F 6C 20 65 64 6F 63 6F 72 63 69  00BD
20 73 73 65 63 6F 72 70 20 72 65 64  00C9
20 65 6D 6F 73 20 64 65 67 67 6F 6C  00D5
67 61 73 73 65 6D 20 72 6F 72 72 65  00E1
                   20 3A 29 73 28 65 00ED
                                     00F3     254
                                     00F3     255 XFLDR_COPY_START:                               ; $PUTMSG MSGVEC for start copying log
                           000F 0006 00F3     256              .WORD   6,^XF
                            00741132 00F7     257              .LONG   UETP$_TEXT!STS$K_ERROR
                           0000 0001 00FB     258              .WORD   1,0
                           000000AF' 00FF     259              .ADDRESS XFLDR_LOG
                            007480B1 0103     260              .LONG   UETP$_COPY_LOG
                           0000 0001 0107     261              .WORD   1,0
                           0000006F' 010B     262              .ADDRESS XFLDR_SYS$ERROR_DESC
                                     010F     263
```

```
                              010F   264 XFLDR_COPY_LINE:                          ; $PUTMSG MSGVEC for copying log line
                    0001 0004 010F   265         .WORD   4,^X1
                     007480B9 0113   266         .LONG   UETP$_COPY_LOG_LINE
                    0000 0002 0117   267         .WORD   2,0
                     00000004 011B   268         .LONG   4
                    00000014' 011F   269         .ADDRESS BUFFER_PTR
                              0123   270
                              0123   271 XFLDR_COPY_FINISH:                        ; $PUTMSG MSGVEC for ucode ' 'r log end
                    000F 0003 0123   272         .WORD   3,^XF
                     007480C1 0127   273         .LONG   UETP$_COPY_LOG_ENDED
                    0000 0001 012B   274         .WORD   1,0
                    0000006F' 012F   275         .ADDRESS XFLDR_SYS$ERROR_DESC
                              0133   276
                              0133   277 MODE:                                     ; Run mode logical name
      45 44 4F 4D 0000013B'010E0000' 0133   278         .ASCID  /MODE/
                              013F   279
                              013F   280 CS:
52 44 20 64 61 42 00000147'010E0000' 013F   281         .ASCID  /Bad DR !AC packet DSL of !XL expected !XL/
20 74 65 6B 63 61 70 20 43 41 21 20 014D
65 20 4C 58 21 20 66 6F 20 4C 53 44 0159
   4C 58 21 20 64 65 74 63 65 70 78 0165
                              0170   282
                              0170   283 NO_RMS_AST_TABLE:                         ; List of errors for which...
                    00000000' 0170   284         .LONG   RMS$_BLN                  ; ...RMS cannot deliver an AST...
                    00000000' 0174   285         .LONG   RMS$_BUSY                 ; ...even if one has an ERR= arg
                    00000000' 0178   286         .LONG   RMS$_CDA                  ; Note that we can search table...
                    00000000' 017C   287         .LONG   RMS$_FAB                  ; ...via MATCHC since <31:16>...
                    00000000' 0180   288         .LONG   RMS$_RAB                  ; ...pattern can't be in <15:0>
                     00000014 0184   289 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                              0184   290
                              0184   291 SYS$INPUT:                                ; Name of device from which...
4E 49 24 53 59 53 0000018C'010E0000' 0184   292         .ASCID  /SYS$INPUT/       ; ...the test can be aborted
                     54 55 50 0192
                              0195   293
                              0195   294 INPUT_ITMLST.                             ; $GETDVI arg list for SYS$INPUT
                    0020 0040 0195   295         .WORD   64,DVI$_DEVNAM            ; We need the equivalence name
           00000014'0000001C' 0199   296         .LONG   BUFFER,BUFFER_PTR
                     00000000 01A1   297         .LONG   0                         ; Terminate the list
                              01A5   298
                              01A5   299 CS1:                                      ; Device class and type control string
21 20 42 58 32 21 000001AD'010E0000' 01A5   300         .ASCID  /!2XB !2XB /
                  20 42 58 32 01B3
                              01B7   301
                              01B7   302 CS2:
52 44 20 65 68 54 000001BF'010E0000' 01B7   303         .ASCID  /The DR!AC data rate is !XW which is !AS megabytes per second./
74 61 72 20 61 74 61 64 20 43 41 21 01C5
69 68 77 20 57 58 21 20 73 69 20 65 01D1
65 6D 20 53 41 21 20 73 69 20 68 63 01DD
20 72 65 70 20 73 65 74 79 62 61 67 01E9
         2E 64 6E 6F 63 65 73 01F5
                              01FC   304
                              01FC   305 CS3:                                      ; Device class-only control string
2A 20 42 58 32 21 00000204'010E0000' 01FC   306         .ASCID  /!2XB **/
                           2A 020A
                              020B   307
                              020B   308 CS4
                    000000C4' 020B   309         .LONG   CS1L
```

```
                          00000213'  020F   310          .ADDRESS .+4
63 20 74 65 6B 63 61 70 20 64 61 42  0213   311          .ASCII  \Bad packet count, expected !UL packets but received !UL.!/\
74 63 65 70 78 65 20 2C 74 6E 75 6F  021F
65 6B 63 61 70 20 4C 55 21 20 64 65  022B
69 65 63 65 72 20 74 75 62 20 73 74  0237
      2F 21 2E 4C 55 21 20 64 65 76  0243
20 67 6E 69 73 73 69 4D 5F 21 5F 21  024D   312          .ASCII  \!_!_Missing !UL!- packet!%S : ! (AC)\
74 65 6B 63 61 70 20 2D 21 4C 55 21  0259
29 43 41 28 23 21 20 3A 20 53 25 21  0265
                          000000C4   0271   313          CS1L=.-CS1-8
                                     0271   314
                                     0271   315 ULOAD_FAILED:
6F 63 75 20 52 44 00000279'010E0000' 0271   316          .ASCID  \DR ucode load failure.\
6C 69 61 66 20 64 61 6F 6C 20 65 64  027F
                        2E 65 72 75  028B
                                     028F   317
                                     028F   318 START_DATA_FAILED:
64 65 6C 69 61 46 00000297'010E0000' 028F   319          .ASCID  \Failed to start data transfer\
61 64 20 74 72 61 74 73 20 6F 74 20  029D
   72 65 66 73 6E 61 72 74 20 61 74  02A9
                                     02B4   320
                                     02B4   321 CNTRLCMSG:
65 74 72 6F 62 41 000002BC'010E0000' 02B4   322          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64  02C2
               43 2F 4C 52 54 43 20  02CE
                                     02D5   323
                                     02D5   324 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000002DD'010E0000' 02D5   325          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74  02E3
                  2E 64 65 69 66 69  02EF
                                     02F5   326
                                     02F5   327 DEAD_CTRLNAME:
20 74 27 6E 61 43 000002FD'010E0000' 02F5   328          .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74  0303
72 61 6D 20 2C 53 41 21 20 72 65 6C  030F
61 73 75 6E 75 20 73 61 20 64 65 6B  031B
4E 49 54 45 55 20 6E 69 20 65 6C 62  0327
         2E 54 41 44 2E 56 45 44 49  0333
                                     033C   329
                                     033C   330 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000344'010E0000' 033C   331          .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74  034A
2E 67 6E 69 74 73 65 74 20 72 6F 66  0356
                                     0362   332
                                     0362   333 ILLEGAL_REC:
61 67 65 6C 6C 49 0000036A'010E0000' 0362   334          .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C  0370
20 65 6C 69 66 20 6E 69 20 74 61 6D  037C
41 44 2E 56 45 44 49 4E 49 54 45 55  0388
                              21 54  0394
                                     0396   335
                                     0396   336 PASS_MSG:
66 6F 20 64 6E 45 0000039E'010E0000' 0396   337          .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20  03A4
61 72 65 74 69 20 4C 55 21 20 68 74  03B0
44 25 21 20 74 61 20 73 6E 6F 69 74  03BC
                                 2E  03C8
```

```
                                    03C9    338
                                    03C9    339 DR780:                                  ; DR780 name
                       30 38 37 00' 03C9    340         .ASCIC  /780/
                                 03 03C9
                                    03CD    341 DR750:                                  ; DR750 name
                       30 35 37 00' 03CD    342         .ASCIC  /750/
                                 03 03CD
                                    03D1    343
                                    03D1    344 INIDEV_UPDERR:                          ; Error during exit nandler
20 72 6F 72 72 45 000003D9'010E0000' 03D1    345         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75  03DF
   2E 54 41 44 2E 56 45 44 49 4E 49  03EB
                                    03F6    346
                                    03F6    347 THREEMIN:                               ; 3 minute delta time
               FFFFFFFF 94B62E00    03F6    348         .LONG   -10*1000*1000*180,-1
                                    03FE    349 ONEMIN:                                 ; 1 minute delta time
               FFFFFFFF DC3CBA00    03FE    350         .LONG   -10*1000*1000*60,-1
                                    0406    351 TENSEC:
               FFFFFFFF FA0A1F00    0406    352         .LONG   -10*1000*1000*10,-1     ; 10 second delta time
                                    040E    353
                                    040E    354 UNIT_DESC:                              ; Descriptor used to convert unit #
                          00000005  040E    355         .LONG   5
                          00000022' 0412    356         .ADDRESS BUFFER+6
                                    0416    357
                                    0416    358 CONT_DESC:                              ; Descriptor used to convert controller...
                         0000 0028  0416    359         .WORD   REC_SIZE,0              ; ...from lowercase to uppercase
                          0000001C' 041A    360         .ADDRESS BUFFER
                                    041E    361
                                    041E    362 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 00000426'010E0000' 041E    363         .ASCID  /file/
                                    042A    364
                                    042A    365 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000432'010E0000' 042A    366         .ASCID  /record/
                                    0438    367
                                    0438    368 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 00000440'010E0000' 0438    369         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53  0446
               44 41 21 20 65 6C 69  0452
                                    0459    370
                                    0459    371 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43  0459    372         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65  0465
                                 20  0471
                          00000019  0472    373         PMTSIZ = .-PROMPT
                                    0472    374
                                    0472    375 BADQUE:
75 71 20 64 61 42 0000047A'010E0000' 0472    376         .ASCID  /Bad queue entry detected! Fatal error!/
65 64 20 79 72 74 6E 65 20 65 75 65  0480
61 74 61 46 20 21 64 65 74 63 65 74  048C
         21 72 6F 72 72 65 20 6C    0498
                                    04A0    377
                                    04A0    378 TEST_HUNG:
6E 75 68 20 52 44 000004A8'010E0000' 04A0    379         .ASCID  /DR hung, check backplane jumpers needed for testing./
63 61 62 20 6B 63 65 68 63 20 2C 67  04AE
65 70 6D 75 6A 20 65 6E 61 6C 70 6B  04BA
6F 66 20 64 65 64 65 65 6E 20 73 72  04C6
   2E 67 6E 69 74 73 65 74 20 72    04D2
```

```
                           04DC   380
                           04DC   381 CMDBLKDES:
                000012D8'  04DC   382         .ADDRESS CMDBLK                 ; LKWSET parameter list for locking
                000015BC'  04E0   383         .ADDRESS CMDBLKEND              ; Down the command block
                           04E4   384
                           04E4   385 NAME_TBL:                               ; Table of pointers to ASCIC packet names
                           04E4   386                                         ; And expected packet status returns
                0000056C   04E4   387         .BLKL   NO_OF_POS_PKTS*2        ; ** NOTE ** table must be in this order
                           056C   388
                           056C   389 PKT_TBL:                                ; Table of pointers to DR packets
                00001598'  056C   390         .ADDRESS FREE_PKT               ; ** NOTE ** table must be in this order
                000012F0'  0570   391         .ADDRESS NOOP_PKT
                00001330'  0574   392         .ADDRESS SET_SELF_PKT
                00001370'  0578   393         .ADDRESS DIAG_WRI_PKT
                000013B0'  057C   394         .ADDRESS READ_DDI_PKT
                00001390'  0580   395         .ADDRESS DIAG_REA_PKT
                000013D0'  0584   396         .ADDRESS WRITE_CH_PKT
                00001400'  0588   397         .ADDRESS WRITE_PKT
                000014A0'  058C   398         .ADDRESS READ_CHA_PKT
                000014D0'  0590   399         .ADDRESS READ_PKT
                00001570'  0594   400         .ADDRESS DIAG_WRT_PKT
                00001350'  0598   401         .ADDRESS CLR_SELF_PKT
                00001310'  059C   402         .ADDRESS HALT_PKT
                           05A0   403
                000004E4   05A0   404         PC1...=NAME_TBL                         ; Set name pointer
                000005A0   05A0   405         PC2...=.                                ; Set name table pointer
                           05A0   406
                           05A0   407         .LIST MEB
                           05A0   408         BUILD READ,0^^00023                     ; Build the tables and the type name
                000004E4   05A0                       .=PC1...            ; PC of ASCIC pkt READ table
                000005A0'  04E4                       .ADDRESS PC2...     ; ASCIC pkt address
                00000023   04E8                       .LONG   ^X00000023  ; Expected packet return code
                000005A0   04EC                       .-^C2...            ; Point to the next ASCIC msg
                           05A0           READ:
      20 2C 44 41 45 52 00'  05A0                     .ASCIC /READ, /
                        06   05A0
                           05A7   409         BUILD READ_CHAIN,00000023
                000004EC   05A7                       .=PC1...            ; PC of ASCIC pkt READ_CHAIN table
                000005A7'  04EC                       .ADDRESS PC2...     ; ASCIC pkt address
                00000023   04F0                       .LONG   ^X00000023  ; Expected packet return code
                000005A7   04F4                       .=PC2...            ; Point to the next ASCIC msg
                           05A7           READ_CHAIN:
2C 4E 49 41 48 43 5F 44 41 45 52 00'  05A7            .ASCIC /READ_CHAIN, /
                        20   05B3
                        0C   05A7
                           05B4   410         BUILD WRITE,00000023
                000004F4   05B4                       .=PC1...            ; PC of ASCIC pkt WRITE table
                000005B4'  04F4                       .ADDRESS PC2...     ; ASCIC pkt address
                00000023   04F8                       .LONG   ^X00000023  ; Expected packet return code
                000005B4   04FC                       .=PC2...            ; Point to the next ASCIC msg
                           05B4           WRITE:
   20 2C 45 54 49 52 57 00'  05B4                     .ASCIC /WRITE, /
                        07   05B4
                           05BC   411         BUILD WRITE_CHAIN,00000023
                000004FC   05BC                       .=PC1...            ; PC of ASCIC pkt WRITE_CHAIN table
                000005BC'  04FC                       .ADDRESS PC2...     ; ASCIC pkt address
                00000023   0500                       .LONG   ^X00000023  ; Expected packet return code
```

```
                      000005BC  0504                       .=PC2...                      ; Point to the next ASCIC msg
                                05BC         WR. __CHAIN:
4E 49 41 48 43 5F 45 54 49 52 57 00' 05BC                  .ASCIC /WRITE_CHAIN, /
                           20 2C  05C8
                              0D  05BC
                                  05CA  412        BUILD WRITE_DEV_CNTRL,00000023
                      00000504  05CA                       .=PC1...                      ; PC of ASCIC pkt WRITE_DEV_CNTRL table
                      000005CA' 0504                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000023  0508                       .LONG   ^X00000023            ; Expected packet return code
                      000005CA  050C                       .=PC2...                      ; Point to the next ASCIC msg
                                05CA         WRITE_DEV_CNTRL:
43 5F 56 45 44 5F 45 54 49 52 57 00' 05CA                  .ASCIC /WRITE_DEV_CNTRL, /
               20 2C 4C 52 54 4E  05D6
                              11  05CA
                                  05DC  413        BUILD RESERVED,0000C000
                      0000050C  05DC                       .=PC1...                      ; PC of ASCIC pkt RESERVED table
                      000005DC' 050C                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000000  0510                       .LONG   ^X00000000            ; Expected packet return code
                      000005DC  0514                       .=PC2...                      ; Point to the next ASCIC msg
                                05DC         RESERVED:
   20 2C 44 45 56 52 45 53 45 52 00' 05DC                  .ASCIC /RESERVED, /
                              0A  05DC
                                  05E7  414        BUILD SET_SELF_TEST,00000023
                      00000514  05E7                       .=PC1...                      ; PC of ASCIC pkt SET_SELF_TEST table
                      000005E7' 0514                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000023  0518                       .LONG   ^X00000023            ; Expected packet return code
                      000005E7  051C                       .=PC2...                      ; Point to the next ASCIC msg
                                05E7         SET_SELF_TEST:
45 54 5F 46 4C 45 53 5F 54 45 53 00' 05E7                  .ASCIC /SET_SELF_TEST, /
                     20 2C 54 53  05F3
                              0F  05E7
                                  05F7  415        BUILD CLR_SELF_TEST,00000003
                      0000051C  05F7                       .=PC1...                      ; PC of ASCIC pkt CLR_SELF_TEST table
                      000005F7' 051C                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000003  0520                       .LONG   ^X00000003            ; Expected packet return code
                      000005F7  0524                       .=PC2...                      ; Point to the next ASCIC msg
                                05F7         CLR_SELF_TEST:
45 54 5F 46 4C 45 53 5F 52 4C 43 00' 05F7                  .ASCIC /CLR_SELF_TEST, /
                     20 2C 54 53  0603
                              0F  05F7
                                  0607  416        BUILD NOOP,00000003
                      00000524  0607                       .=PC1...                      ; PC of ASCIC pkt NOOP table
                      00000607' 0524                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000003  0528                       .LONG   ^X00000003            ; Expected packet return code
                      00000607  052C                       .=PC2...                      ; Point to the next ASCIC msg
                                0607         NOOP:
            20 2C 50 4F 4F 4E 00' 0607                     .ASCIC /NOOP, /
                              06  0607
                                  060E  417        BUILD DIAG_READ_INT,00000023
                      0000052C  060E                       .=PC1...                      ; PC of ASCIC pkt DIAG_READ_INT table
                      0000060E' 052C                       .ADDRESS PC2...               ; ASCIC pkt address
                      00000023  0530                       .LONG   ^X00000023            ; Expected packet return code
                      0000060E  0534                       .=PC2...                      ; Point to the next ASCIC msg
                                060E         DIAG_READ_INT:
49 5F 44 41 45 52 5F 47 41 49 44 00' 060E                  .ASCIC /DIAG_READ_INT, /
                     20 2C 54 4E  061A
                              0F  060E
```

UE
V0

```
                                  061E   418         BUILD DIAG_WRIT_INT,00000023
                        00000534  061E                       .=PC1...                 ; PC of ASCIC pkt DIAG_WRIT_INT table
                        0000061E' 0534                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000023  0538                       .LONG    ^X00000023      ; Expected packet return code
                        0000061E  053C                       .=PC2...                 ; Point to the next ASCIC msg
                                  061E          DIAG_WRIT_INT:
49 5F 54 49 52 57 5F 47 41 49 44 00' 061E                    .ASCIC /DIAG_WRIT_INT, /
                      20 2C 54 4E  062A
                                0F  061E
                                  062E   419         BUILD READ_DDI,00000023
                        0000053C  062E                       .=PC1...                 ; PC of ASCIC pkt READ_DDI table
                        0000062E' 053C                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000023  0540                       .LONG    ^X00000023      ; Expected packet return code
                        0000062E  0544                       .=PC2...                 ; Point to the next ASCIC msg
                                  062E          READ_DDI:
   20 2C 49 44 44 5F 44 41 45 52 00' 062E                    .ASCIC /READ_DDI, /
                                0A  062E
                                  0639   420         BUILD DIAG_WRT_CNTRL,00000023
                        00000544  0639                       .=PC1...                 ; PC of ASCIC pkt DIAG_WRT_CNTRL table
                        00000639' 0544                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000023  0548                       .LONG    ^X00000023      ; Expected packet return code
                        00000639  054C                       .=PC2...                 ; Point to the next ASCIC msg
                                  0639          DIAG_WRT_CNTRL:
4E 43 5F 54 52 57 5F 47 41 49 44 00' 0639                    .ASCIC /DIAG_WRT_CNTRL, /
                   20 2C 4C 52 54  0645
                                10  0639
                                  064A   421         BUILD SET_RAND_ENABLE,00000000
                        0000054C  064A                       .=PC1...                 ; PC of ASCIC pkt SET_RAND_ENABLE table
                        0000064A' 054C                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000000  0550                       .LONG    ^X00000000      ; Expected packet return code
                        0000064A  0554                       .=PC2...                 ; Point to the next ASCIC msg
                                  064A          SET_RAND_ENABLE:
4E 45 5F 44 4E 41 52 5F 54 45 53 00' 064A                    .ASCIC /SET_RAND_ENABLE, /
                20 2C 45 4C 42 41  0656
                                11  064A
                                  065C   422         BUILD CLR_RAND_ENABLE,00000000
                        00000554  065C                       .=PC1...                 ; PC of ASCIC pkt CLR_RAND_ENABLE table
                        0000065C' 0554                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000000  0558                       .LONG    ^X00000000      ; Expected packet return code
                        0000065C  055C                       .=PC2...                 ; Point to the next ASCIC msg
                                  065C          CLR_RAND_ENABLE:
4E 45 5F 44 4E 41 52 5F 52 4C 43 00' 065C                    .ASCIC /CLR_RAND_ENABLE, /
                20 2C 45 4C 42 41  0668
                                11  065C
                                  066E   423         BUILD HALT,00000003
                        0000055C  066E                       .=PC1...                 ; PC of ASCIC pkt HALT table
                        0000066E' 055C                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000003  0560                       .LONG    ^X00000003      ; Expected packet return code
                        0000066E  0564                       .=PC2...                 ; Point to the next ASCIC msg
                                  066E          HALT:
            20 2C 54 4C 41 48 00' 066E                       .ASCIC /HALT, /
                                06  066E
                                  0675   424         BUILD FREE,00000029
                        00000564  0675                       .=PC1...                 ; PC of ASCIC pkt FREE table
                        00000675' 0564                       .ADDRESS PC2...          ; ASCIC pkt address
                        00000029  0568                       .LONG    ^X00000029      ; Expected packet return code
                        00000675  056C                       .=PC2...                 ; Point to the next ASCIC msg
```

```
                         0675        FREE:
  20 2C 45 45 52 46 00'  0675                        .ASCIC /FREE, /
                     06  0675
                         067C   425          .NLIST MEB
                         067C          .SBTTL  Read/Write Data
                     00000000          .PSECT  RWDATA,WRT,NOEXE,PAGE
                         0000
                         0000  TTCHAN:                                          ; Channel associated with ctrl. term.
                0000     0000          .WORD   0
                         0002
                         0002  MBCHAN:                                          ; Mailbox channel
                0000     0002          .WORD   0
                         0004
                         0004  CHAN:                                            ; DR780 channel
                0000     0004          .WORD   0
                         0006
                         0006  PID:                                             ; PID storage for ucode load process
            00000000     0006          .LONG   0
                         000A
                         000A  FLAG:                                            ; Miscellaneous flag bits
                0000     000A          .WORD   0                                ; (See Equated Symbols for definitions)
                         000C
                         000C  FAO_BUF:                                         ; FAO output string descriptor
           0000 0084     000C          .WORD   TEXT_BUFFER,0
           0000001C'     0010          .ADDRESS BUFFER
                         0014
                         0014  BUFFER_PTR:                                      ; Fake .ASCID buffer for misc. strings
           0000 0084     0014          .WORD   TEXT_BUFFER,0                    ; A word for length, a word for desc.
           0000001C'     0018          .ADDRESS BUFFER
                         001C
                         001C  BUFFER:                                          ; FAO output and other misc. buffer
            000000A0     001C          .BLKB   TEXT_BUFFER
                         00A0
                         00A0  DEVDSC:                                          ; Device name descriptor
           0000 000A     00A0          .WORD   MAX_DEV_DESIG,0
           000000BF'     00A4          .ADDRESS DEV_NAME
                         00A8
                         00A8  PROCESS_NAME:                                    ; Process name
38 37 52 44 000000B0'010E0000'  00A8   .ASCID  /DR78/
            0000000B     00B4          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
            000000BF     00B4          .BLKB   PROCESS_NAME_FREE
                         00BF
                         00BF  DEV_NAME:                                        ; Device name buffer
            000000CE     00BF          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
            0000000F     00CE          NAME_LEN = .-DEV_NAME
                         00CE
                         00CE  DIB:                                             ; Device Information Block
           0000 0074     00CE          .WORD   DIB$K_LENGTH,0
           000000D6'     00D2          .ADDRESS DIBBUF
                         00D6  DIBBUF:
            0000014A     00D6          .BLKB   DIB$K_LENGTH
                         014A
                         014A  ERROR_COUNT:                                     ; Cumulative error count at runtime
            00000000     014A          .LONG   0
                         014E
                         014E  STATUS:                                          ; Status value on program exit
            00000000     014E          .LONG   0
                         0152
                         0152  QUAD_STATUS:                                     ; IO status block for misc sys. svcs.
   00000000 00000000     0152          .QUAD   0
                         015A
                         015A  INADDRESS:                                  ; $CRMPSC address storage
00000000 00000000        015A          .LONG   0,0
                         0162
                         0162  OUTADDRESS:
00000000 00000000        0162          .LONG   0,0
                         016A
                         016A  UNIT_NUMBER:                                ; Current dev unit number
             0000        016A          .WORD   0
                         016C
                         016C  DEVNAM_LEN:                                 ; Current device name length
             0000        016C          .WORD   0
                         016E
                         016E  RANDOM1:                                    ; Random word #1
         AAAAAAAA        016E          .LONG   ^XAAAAAAAA
                         0172
                         0172  RANDOM2:                                    ; Random word #2
         A72EA72E        0172          .LONG   ^XA72EA72E
                         0176
                         0176  ITERATION:                                  ; # of times all tests were executed
         00000000        0176          .LONG   0
                         017A
                         017A  PASS:                                       ; Pass count
         00000000        017A          .LONG   0
                         017E
                         017E  MSG_BLOCK:                                  ; Auxiliary $GETMSG info
         00000182        017E          .BLKB   4
                         0182
                         0182  EXIT_DESC:                                  ; Exit handler descriptor
         00000000        0182          .LONG   0
         00000CF0'       0186          .ADDRESS EXIT_HANDLER
         00000001        018A          .LONG   1
         0000014E'       018E          .ADDRESS STATUS
                         0192
                         0192  ARG_COUNT:                                  ; Argument counter used by ERROR_EXIT
         00000000        0192          .LONG   0
                         0196
                         0196  ;
                         0196  ; Head of self-relative UETP unit block queue.
                         0196  ;
                         0196          .ALIGN QUAD
                         0198
                         0198  UNIT_LIST:                                  ; Head of unit block circular list
00000000 00000000        0198          .QUAD   0
                         01A0
                         01A0  NEW_NODE:                                   ; Newly aquired node address
00000000 00000000        01A0          .QUAD   0
                         01A8
                         01A8  PKT_CNT:
         00000000        01A8          .LONG   0                           ; Cumulative packet count for this PKT_CHECK
                         01AC
                         01AC  PACK_REMOVED:
         00000000        01AC          .LONG   0                           ; Bit mask record of the packets which
                         01B0                                              ; Have been removed from the termination
                         01B0                                              ; Queue. Bit position is directly related
                         01B0                                              ; To the fuction code e.g. the READ bit
                         01B0                                              ; Is 0 and the READ_CHAIN bit is 1
                         01B0  ARGS:
```

```
         00000200  01B0          .BLKL   20                      ; Space for 20 arguments
                   0200
                   0200  BADRPKT:
         00000000  0200          .LONG   0                       ; Bad DR packet message desc.
         0000001C' 0204          .ADDRESS BUFFER
                   0208
                   0208  RATE_BUF:
         00000007  0208          .LONG   7                       ; Buffer for ASCII rate in decimal
         00000210' 020C          .ADDRESS .+4
         00000217  0210          .BLKB   7
                   0217
                   0217  RATE_FLOAT:
00000000 00000000  0217          .DOUBLE 0                       ; Storage for double format rate
                   021F
                   021F  RATE_DESC:
         00000000  021F          .LONG   0                       ; Desc for the full rate message
         0000001C' 0223          .ADDRESS BUFFER
                   0227
                   0227  DRIOSTAT:
00000000 00000000  0227          .QUAD   0                       ; DR780 IO status block
                   022F
                   022F  BUFBLK:
                   022F  TEST_DATA:
                   022F  OUTPUT_BUF:                             ; Primary output data buffer
         00000A2F  022F          .BLKB   BUFSIZ
                   0A2F
                   0A2F  INPUT_BUF:                              ; Primary input data buffer
         0000122F  0A2F          .BLKB   BUFSIZ
                   122F
                   122F  INPUT1_BUF:                             ; Secondary input data buffer
         000012AF  122F          .BLKB   128
         00001080  12AF          BUFBLKSIZ=.-BUFBLK
                   12AF
                   12AF  ;
                   12AF  ; Data transfer command table
                   12AF  ;
                   12AF  CMDTBL:
         000002E4' 12AF          .LONG   CMDBLKSIZ               ; Length of command block in bytes
         000012D8' 12B3          .LONG   CMDBLK                  ; Address of start of command block.
                   12B7                                          ; Also head of input queue
         00001080  12B7          .LONG   BUFBLKSIZ               ; Length of buffer block in bytes
         0000022F' 12BB          .LONG   BUFBLK                  ; Base address of the buffer block
         00000878' 12BF          .LONG   PKT1_AST                ; Address of the packet interrupt routine
         00000000  12C3          .LONG   0                       ; Parameter to be passed to AST routine.
               00  12C7          .BYTE   0                       ; Data transfer rate
               03  12C8          .BYTE   XF$M_CMT_DIPEAB!-
                   12C9                  XF$M_CMT_SETRTE         ; Flags byte
             0000  12C9          .WORD   0                       ; Not used
         000012CF' 12CB          .LONG   GOBIT                   ; Address into which the address of the
                   12CF                                          ; DR's gobit will be written by QIO.
                   12CF                                          ;
         00000020  12CF  CMDTBLSIZ=.-CMDTBL                      ; Define the length of the command table
                   12CF  ;
                   12CF  ;Long word to receive the address of the gobit
                   12CF  ;
                   12CF  GOBIT:
         00000000  12CF          .LONG   0
```

```
                 12D3          .Subtitle COMMAND BLOCK & PACKETS
                 12D3   ;
                 12D3   ; This is the start of the command block from which the DR fetches its commands
                 12D3   ;
                 12D3   ; The commands are in the form of a block of memory called a packet which is
                 12D3   ; linked into a list using the interlocked queue instructions.
                 12D3   ; The DR removes the command packets from the INPUT QUEUE processes them
                 12D3   ; and replaces them onto the TERMination QUEUE.
                 12D3   ; A status longword is written into each packet before it is connected to
                 12D3   ; to the TERMQ.
                 12D3   ;
                 12D3   ; The command block must be quad word aligned to support the queue instructions
                 12D3   ;
00'00'00'00'00'  12D3          .ALIGN  QUAD    0
                 12D8   CMDBLK::                                ;
        000012E0 12D8   INPTQH: .BLKQ   1
        000012E8 12E0   TERMQH: .BLKQ   1
        000012F0 12E8   FREEQH: .BLKQ   1
                 12F0   ;
                 12F0   ; Packet to do a nop command.
                 12F0   ;
                 12F0   NOOP_PKT:
        00000000 12F0          .LONG   0                        ; Queue forward link
        00000000 12F4          .LONG   0                        ; Queue backward link
            0000 12F8          .WORD   0                        ; Log area and message length
              08 12FA          .BYTE   XF$K_PKT_NOP@XF$V_PKT_FUNC ; Command = nop
              80 12FB          .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
                 12FC                                           ; Interrupt unconditionally.
        00000000 12FC          .LONG   0                        ; Byte count not used here
        00000000 1300          .LONG   0                        ; Va not used here
        00000000 1304          .LONG   0                        ; Residual memory byte count
                 1308                                           ; Not used here.
        00000000 1308          .LONG   0                        ; Residual ddi byte count
                 130C                                           ; Not used here.
        00000000 130C          .LONG   0                        ; Dr status longword for this pkt
                 1310                                           ;
                 1310   ;
                 1310   ; Packet to do a halt command.
                 1310   ;
                 1310   ; This packet will cause two AST's to be queued regardless of the state
                 1310   ; of the interrupt control field.
                 1310   ;
                 1310   ;
                 1310
                 1310   HALT_PKT:                               ;
        00000000 1310          .LONG   0                        ; Queue forward link
        00000000 1314          .LONG   0                        ; Queue backward link
            0000 1318          .WORD   0                        ; Log area and message length
            000F 131A          .WORD   XF$K_PKT_HALT@XF$V_PKT_FUNC ; Command = halt
                 131C                                           ; Interrupt field ignored here
        00000000 131C          .LONG   0                        ; Byte count not used here
        00000000 1320          .LONG   0                        ; Va not used here
        00000000 1324          .LONG   0                        ; Residual memory byte count
                 1328                                           ; Not used here.
        00000000 1328          .LONG   0                        ; Residual DDI byte count
                 132C                                           ; Not used here.
        00000000 132C          .LONG   0                        ; DR status longword for this pkt
```

```
            1330                                                         ;
            1330       ;
            1330       ; Packet to set self test mode.
            1330       ;
            1330       ; This packet will get error status if DDI DISABLE is set when it is executed.
            1330       ;
            1330      SET_SELF_PKT:                                      ;
            1330                                                         ;
  00000000  1330              .LONG   0                                  ; Queue forward link
  00000000  1334              .LONG   0                                  ; Queue backward link
      0000  1338              .WORD   0                                  ; Log area and message length
        06  133A              .BYTE   XF$K_PKT_SETTST@XF$V_PKT_FUNC ; Command = set self test
        80  133B              .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
  00000000  133C              .LONG   0                                  ; Byte count not used here
  00000000  1340              .LONG   0                                  ; Va not used here
  00000000  1344              .LONG   0                                  ; Residual memory byte count
            1348                                                         ; Not used here.
  00000000  1348              .LONG   0                                  ; Residual DDI byte count
            134C                                                         ; Not used here.
  00000000  134C              .LONG   0                                  ; DR status longword for this pkt
            1350                                                         ;
            1350       ;
            1350       ; Packet to clear self test.
            1350       ;
            1350
            1350      CLR_SELF_PKT:                                      ;
            1350                                                         ;
  00000000  1350              .LONG   0                                  ; Queue forward link
  00000000  1354              .LONG   0                                  ; Queue backward link
      0000  1358              .WORD   0                                  ; Log area and message length
        07  135A              .BYTE   XF$K_PKT_CLRTST@XF$V_PKT_FUNC ; Command = clear self test
        80  135B              .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
  00000000  135C              .LONG   0                                  ; Byte count not used here
  00000000  1360              .LONG   0                                  ; Va not used here
  00000000  1364              .LONG   0                                  ; Residual memory byte count
            1368                                                         ; Not used here.
  00000000  1368              .LONG   0                                  ; Residual DDI byte count
            136C                                                         ; Not used here.
  00000000  136C              .LONG   0                                  ; DR status longword for this pkt
            1370                                                         ;
            1370       ;
            1370       ; Command packet to do a diagnostic write internal
            1370       ;
            1370       ; This command is used to test the dr's internal silo.
            1370       ; The number of bytes specified by the byte count are read from memory
            1370       ; and stored in the silo.
            1370       ;
            1370
            1370      DIAG_WRI_PKT:                                      ;
            1370                                                         ;
  00000000  1370              .LONG   0                                  ; Queue forward link
  00000000  1374              .LONG   0                                  ; Queue backward link
      0000  1378              .WORD   0                                  ; Log area and message length
        0A  137A              .BYTE   XF$K_PKT_DIAGWI@XF$V_PKT_FUNC ; Command = diag write internal
        80  137B              .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
  00000080  137C              .LONG   128                                ; Byte count is 128 even though
            1380                                                         ; Only 124 bytes are valid
```

```
0000022F' 1380          .ADDRESS OUTPUT_BUF                 ; Address of data buffer
00000000  1384          .LONG   0                           ; Residual memory byte count
00000000  1388          .LONG   0                           ; Residual DDI byte count
00000000  138C          .LONG   0                           ; DR status long word for this pkt
          1390                                              ;
          1390  ;
          1390  ; Command packet to do a diagnostic read internal command.
          1390  ;
          1390  ; This command is used to read the data in the DR's internal silo.
          1390  ; The number of bytes specified by the byte count are read from the dr
          1390  ; and written to memory specified by the virtual address field.
          1390  ;
          1390  ;
          1390
          1390  DIAG_REA_PKT:                               ;
00000000  1390          .LONG   0                           ; Queue forward link
00000000  1394          .LONG   0                           ; Queue backward link
    0000  1398          .WORD   0                           ; Log area and message length
      09  139A          .BYTE   XF$K_PKT_DIAGRI@XF$V_PKT_FUNC ; Command = diag read internal
      00  139B          .BYTE   XF$K_PKT_UNCOND@XF$V_PKT_INTCTL ; Interrupt unconditionally
00000080  139C          .LONG   128                         ; Byte count is 128 even though
          13A0                                              ; Only 124 bytes are valid
0000122F' 13A0          .ADDRESS INPUT1_BUF                 ; Address of data buffer
00000000  13A4          .LONG   0                           ; Residual memory byte count
00000000  13A8          .LONG   0                           ; Residual DDI byte count
00000000  13AC          .LONG   0                           ; DR status long word for this pkt
          13B0                                              ;
          13B0  ;
          13B0  ; Command packet to do a diagnostic read DDI command
          13B0  ;
          13B0  ; This command wraps the data around on the DDI bus and stores it back into
          13B0  ; the silo.
          13B0  ;
          13B0  ;
          13B0  READ_DDI_PKT:                               ;
          13B0                                              ;
00000000  13B0          .LONG   0                           ; Queue forward link
00000000  13B4          .LONG   0                           ; Queue backward link
    0000  13B8          .WORD   0                           ; Log area and message length
      0B  13BA          .BYTE   XF$K_PKT_DIAGRD@XF$V_PKT_FUNC ; Command = diag read DDI
      80  13BB          .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL  ; No interrupt.
00000080  13BC          .LONG   128                         ; Byte count = silo size
00000000  13C0          .LONG   0                           ; Address field not used
00000000  13C4          .LONG   0                           ; Residual memory byte count
00000000  13C8          .LONG   0                           ; Residual DDI byte count
00000000  13CC          .LONG   0                           ; DR status long word for this pkt
          13D0                                              ;
          13D0  ;
          13D0  ; Packet to do a write chained command.
          13D0  ;
          13D0  WRITE_CH_PKT:                               ;
00000000  13D0          .LONG   0                           ; Queue forward link
00000000  13D4          .LONG   0                           ; Queue backward link
    0010  13D8          .WORD   16                          ; Log area and message length
      03  13DA          .BYTE   XF$K_PKT_WRTCHN@XF$V_PKT_FUNC ; Command = write chained
      98  13DB          .BYTE   <XF$K_PKT_CBDMBC@XF$V_PKT_CISEL>!-
          13DC                  <XF$K_PKT_NOINT@XF$V_PKT_INTCTL> ; No interrupt. Send command,
          13DC                                              ; Byte count,device message
          13DC                                              ; Message is 16 bytes long
0000003B  13DC            .LONG   59                        ; Byte count is 59 to keep
          13E0                                              ; Things on odd boundries.
0000022F' 13E0            .ADDRESS OUTPUT_BUF               ; Address of data buffer
00000000  13E4            .LONG   0                         ; Residual memory byte count
00000000  13E8            .LONG   0                         ; Residual DDI byte count
00000000  13EC            .LONG   0                         ; DR status long word for this pkt
          13F0                                              ; Generate and incrementing pattern
          13F0                                              ; For the device message.
00000000  13F0            X=0
          13F0            .REPT   16                        ; Device message is 16 bytes long
          13F0            .BYTE X
          13F0            X=X+1
      00  13F0            .ENDR
          1400                                              ;
          1400  ;
          1400  ; Command packet do to a write device command
          1400  ;
          1400  WRITE_PKT:                                  ;
00000000  1400            .LONG   0                         ; Queue forward link
00000000  1404            .LONG   0                         ; Queue backward link
    0080  1408            .WORD   128                       ; Log area and message length
      02  140A            .BYTE   XF$K_PKT_WRT@XF$V_PKT_FUNC ; Command = write device
      98  140B            .BYTE   <XF$K_PKT_CBDMBC@XF$V_PKT_CISEL>!-
          140C                    <XF$K_PKT_NOINT@XF$V_PKT_INTCTL> ; No interrupt. Send command,
          140C                                              ; Byte count,device message
          140C                                              ; Message is 128 bytes long
000007C5  140C            .LONG   1989                      ; Byte count is 1989 to keep
          1410                                              ; Things on odd boundries.
0000026A' 1410            .ADDRESS OUTPUT_BUF+59            ; Address of data buffer
00000000  1414            .LONG   0                         ; Residual memory byte count
00000000  1418            .LONG   0                         ; Residual DDI byte count
00000000  141C            .LONG   0                         ; DR status long word for this pkt
          1420                                              ; Device message for this packet
          1420                                              ; Even though in self test mode
          1420                                              ; The dr will not look at the message
          1420                                              ; An incrementing pattern is used.
000000FF  1420            X=^XFF
          1420            .REPT   128                       ; Generate an decrementing pattern
          1420            .BYTE   X
          1420            X=X-1
      FF  1420            .ENDR
          14A0                                              ;
          14A0  ;
          14A0  ; Command packet to do a read chained command.
          14A0  ;
          14A0  ; This packet must only be executed in self test mode.
          14A0  ; A device message is transmitted to never never land to use more
          14A0  ; microcode in the DR.
          14A0  ;
          14A0  READ_CHA_PKT:                               ;
00000000  14A0            .LONG   0                         ; Queue forward link
00000000  14A4            .LONG   0                         ; Queue backward link
    0010  14A8            .WORD   16                        ; Log area and message length
      01  14AA            .BYTE   XF$K_PKT_RDCHN@XF$V_PKT_FUNC ; Command = read chained
      98  14AB            .BYTE   <XF$K_PKT_CBDMBC@XF$V_PKT_CISEL>!-
```

```
          14AC                    <XF$K_PKT_NOINT@XF$V_PKT_INTCTL> ; No interrupt device message
          14AC                                                     ; 16 bytes long.send command
          14AC                                                     ; Byte count and device message
          14AC
0000003B  14AC            .LONG   59                       ; Byte count is 59 to keep
          14B0                                             ; Things on odd boundries.
00000A2F' 14B0            .ADDRESS INPUT_BUF               ; Address of data buffer
00000000  14B4            .LONG   0                        ; Residual memory byte count
00000000  14B8            .LONG   0                        ; Residual DDI byte count
00000000  14BC            .LONG   0                        ; DR status long word for this pkt
          14C0                                             ; Generate and incrementing pattern
          14C0                                             ; For the device message.
00000000  14C0            X=0
          14C0            .REPT   16                       ; Device message is 16 bytes long
          14C0            .BYTE X
          14C0            X=X+1
      00  14C0            .ENDR
          14D0                                             ;
          14D0    ;
          14D0    ; Command packet do to a read device command
          14D0    ;
          14D0    READ_PKT:                                ;
00000000  14D0            .LONG   0                        ; Queue forward link
00000000  14D4            .LONG   0                        ; Queue backward link
    0080  14D8            .WORD   128                      ; Log area and message length
      00  14DA            .BYTE   XF$K_PKT_RD@XF$V_PKT_FUNC ; Command = read device
      18  14DB            .BYTE   <XF$K_PKT_CBDMBC@XF$V_PKT_CISEL>!-
          14DC                    <XF$K_PKT_UNCOND@XF$V_PKT_INTCTL> ; Interrupt when done,send command
          14DC                                                      ; Byte count and device message.
          14DC                                                      ; Device message is 128 bytes.
000007C5  14DC            .LONG   1989                     ; Byte count is 1989 to keep
          14E0                                             ; Things on odd boundries.
00000A6A' 14E0            .ADDRESS INPUT_BUF+59            ; Address of data buffer
00000000  14E4            .LONG   0                        ; Residual memory byte count
00000000  14E8            .LONG   0                        ; Residual DDI byte count
00000000  14EC            .LONG   0                        ; DR status long word for this pkt
          14F0                                             ; Device message for this packet
          14F0                                             ; Even though in self test mode
          14F0                                             ; The dr will not look at the message
          14F0                                             ; An incrementing pattern is used.
00000000  14F0            X=0
          14F0            .REPT   128                      ; Generate an incrementing pattern
          14F0            .BYTE   X
          14F0            X=X+1
      00  14F0            .ENDR
          1570                                             ;
          1570    ;
          1570    ; Command packet to do a diagnostic write device message command.
          1570    ;
          1570    ; This command writes a single byte onto the control bus and reads it
          1570    ; back again. A packet is then removed from FREE Q and the data read is
          1570    ; placed into the message area of this packet.
          1570    ;
          1570    ; This command can only be executed in self test mode.
          1570    ;
          1570    DIAG_WRT_PKT:                            ;
          1570                                             ;
```

```
00000000  1570   884            .LONG   0                           ; Queue forward link
00000000  1574   885            .LONG   0                           ; Queue backward link
    0001  1578   886            .WORD   1                           ; Log area and message length
    000C  157A   887            .WORD   XF$K_PKT_DIAGWC@XF$V_PKT_FUNC ; Command = diagnostic write
      80  157C   888            .BYTE   XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
          157D   889                                                ;
00000000  157D   890            .LONG   0                           ; Byte count not used here
00000000  1581   891            .LONG   0                           ; Va field not used here
00000000  1585   892            .LONG   0                           ; Residual memory byte count
          1589   893                                                ; Not used here
00000000  1589   894            .LONG   0                           ; Residual DDI byte count
          158D   895                                                ; Not used here.
00000000  158D   896            .LONG   0                           ; DR status longword for this pkt.
          1591   897    DIAG_CNTRL_MESS:
000000AA  1591   898            .LONG   ^X0AA                       ; Longword for the device message
          1595   899                                                ; This is modified dynamically
          1595   900                                                ;
          1595   901    ;
          1595   902    ; Packet to be placed onto FREE QUEUE to receive the message from the
          1595   903    ; diagnostic write device message command.
          1595   904    ;
          1595   905    ;
          1595   906            .ALIGN  QUAD
          1598   907    FREE_PKT:                                   ;
          1598   908                                                ;
00000000  1598   909            .LONG   0                           ; Queue forward link
00000000  159C   910            .LONG   0                           ; Queue backward link
00000001  15A0   911            .LONG   1                           ; Reserve 1 byte for the incoming
          15A4   912                                                ; Message
00000000  15A4   913            .LONG   0                           ; Byte count not used here
00000000  15A8   914            .LONG   0                           ; Va not used here
00000000  15AC   915            .LONG   0                           ; Residual byte counts not used here
00000000  15B0   916            .LONG   0                           ;
00000000  15B4   917            .LONG   0                           ; DR status longword for this packet.
00000000  15B8   918            .LONG   0                           ; Long word to receive the message byte.
          15BC   919                                                ;
          15BC   920                                                ;
          15BC   921    ;
          15BC   922    ; End of command block
          15BC   923    ;
000002E4  15BC   924            CMDBLKSIZ=.-CMDBLK                  ; Define the length of the
          15BC   925    CMDBLKEND:                                  ; Command block
```

```
                  15BC          .SBTTL  RMS-32 Data Structures
                  15BC          .ALIGN  LONG
                  15BC
                  15BC    SYSIN_FAB:                                    ; Allocate FAB for SYS$INPUT
                  15BC          $FAB-
                  15BC          FNM = <SYS$INPUT>
                  160C
                  160C    SYSIN_RAB:                                    ; Allocate RAB for SYS$INPUT
                  160C          $RAB-
                  160C          FAB = SYSIN_FAB -
                  160C          ROP = PMT,-
                  160C          PBF = PROMPT,-
                  160C          PSZ = PMTSIZ,-
                  160C          UBF = DEV_NAME,-
                  160C          USZ = NAME_LEN
                  1650
                  1650    INI_FAB:                                      ; Allocate FAB for UETINIDEV
                  1650          $FAB-
                  1650          FAC = <GET,PUT,UPD>,-
                  1650          RAT = CR,-
                  1650          SHR = <GET,PUT,UPI>,-
                  1650          FNM = <UETINIDEV.DAT>
                  16A0
                  16A0    INI_RAB:                                      ; Allocate RAB for UETINIDEV
                  16A0          $RAB-
                  16A0          FAB = INI_FAB,-
                  16A0          RBF = BUFFER,-
                  16A0          UBF = BUFFER,-
                  16A0          USZ = REC_SIZE
                  16E4
                  16E4    DDB_RFA:                                      ; RFA storage for INI_RAB
        000016EA  16E4          .BLKB   6
                  16EA
                  16EA          .ALIGN  LONG
                  16EC    SUP_FAB:                                      ; Allocate FAB for UETSUPDEV
                  16EC          $FAB-
                  16EC          FAC = GET,-
                  16EC          SHR = <UPI,GET>,-
                  16EC          RAT = CR,-
                  16EC          FOP = UFO,-
                  16EC          FNM = <UETSUPDEV.DAT>
                  173C
                  173C    ;
                  173C    ; Dummy FAB and RAB to copy to the UETP unit blocks
                  173C    ; The following FAB and RAB must be contiguous and in this order!
                  173C    ;
                  173C
                  173C    DUMMY_FAB:
                  173C          $FAB
                  178C
                  178C    DUMMY_RAB:
                  178C          $RAB    RSZ = WRITE_SIZE,-
                  178C                  USZ = READ_SIZE
                  17D0
                  17D0    XFLDR_SYS$ERROR_FAB:                          ; Gets possible log file from ucode ldr
                  17D0          $FAB    FNS = XFLDR_SYS$ERROR_LENGTH,-
                  17D0                  FNA = XFLDR_SYS$ERROR
```

```
     1820   984
     1820   985 XFLDR_SYS$ERROR_RAB:
     1820   986          $RAB    FAB = XFLDR_SYS$ERROR_FAB,-
     1820   987                  USZ = TEXT_BUFFER,-
     1820   988                  UBF = BUFFER
```

```
                    1864         .SBTTL  Main Program
                00000000         .PSECT  DR78,EXE,NOWRT,PAGE
                    0000
                    0000         .DEFAULT DISPLACEMENT,WORD
                    0000
                    0000   ;+
                    0000   ; This is the main code for the DR780 test. The byte transfer sizes were chosen
                    0000   ; for hardware dependent reasons.  The test sequence is as follows:
                    0000   ;
                    0000   ;        1.      do a NOP packet
                    0000   ;        2.      do a set self test mode packet
                    0000   ;        3.      do a 128. byte write diagnostic internal packet
                    0000   ;        4.      do a 128. byte read diagnostic DDI packet
                    0000   ;        5.      do a 128. byte read diagnostic internal packet
                    0000   ;        6.      do a 59. byte write chained packet
                    0000   ;        7.      do a 1989. byte write packet
                    0000   ;        8.      do a 59. byte read chained packet
                    0000   ;        9.      do a 1989. byte read packet
                    0000   ;        10.     do a 1. byte diagnostic control message packet
                    0000   ;        11.     do a halt packet
                    0000   ;        12.     set the go bit
                    0000   ;        13.     check the 124. bytes transfered by the diag write/read internal
                    0000   ;        14.     check the packet status and packet count
                    0000   ;        15.     check for control C's
                    0000   ;        16.     loop until 3 minutes are up
                    0000   ;
                    0000   ;-
                    0000
              0000  0000   .ENTRY UETDR7800,^M<>                     ; Entry mask
                    0002
6D   0AE7'CF   DE   0002         MOVAL   SSERROR,(FP)                ; Declare exception handler
                    0007         $SETSFM_S ENBFLG = #1               ; Enable system service failure mode
                    0010         $DCLEXH_S DESBLK = EXIT_DESC        ; Declare an exit handler
                    001B
                    001B         $OPEN   FAB = SYSIN_FAB,-           ; Open SYS$INPUT
                    001B                 ERR = RMS_ERROR
                    002A         $CONNECT RAB = SYSIN_RAB,-          ; Connect RAB to SYS$INPUT
                    002A                 ERR = RMS_ERROR
           02  E1   0039         BBC     S^#DEV$V_TRM,-              ; BR if SYS$INPUT is NOT a terminal
  1E 15FC'CF        003B                 SYSIN_FAB+FAB$L_DEV,10$
                    003F         $TRNLOG_S LOGNAM = CONTROLLER,-     ; Allow terminal user to specify...
                    003F                   RSLLEN = DEVNAM_LEN,-     ; ...a logial name...
                    003F                   RSLBUF = DEVDSC           ; ...for the controller to test
     01    50  D1   0058         CMPL    R0,#SS$_NORMAL              ; Was a controller specified?
           2E  13   005B         BEQL    PROC_CONT_NAME              ; BR if it was - go process it
                    005D   10$:
                    005D         $GET    RAB = SYSIN_RAB,-           ; Read SYS$INPUT...
                    005D                 ERR = RMS_ERROR             ; ...for the controller name
     162E'CF   B0   006C         MOVW    SYSIN_RAB+RAB$W_RSZ,-       ; Save the name length
     016C'CF        0070                 DEVNAM_LEN
           16  12   0073         BNEQ    PROC_CONT_NAME              ; BR if we got something
014E'CF    14  D0   0075         MOVL    #SS$_BADPARAM,STATUS        ; Save an exit status if not
     02D5'CF   DF   007A         PUSHAL  NO_CTRLNAME                 ; Prepare for message...
           01  DD   007E         PUSHL   #1                          ; ...arg count
 00741132  8F  DD   0080         PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...signal name
           03  DD   0086         PUSHL   #3                          ; ...arg count
         0BE5  31   0088         BRW     ERROR_EXIT                  ; ...go tell of bad setup
```

```
                                 008B
                                 008B   PROC_CONT_NAME:
  00A0'CF     016C'CF      3C    008B          MOVZWL   DEVNAM_LEN,DEVDSC         ; Set the device name length
              00A0'CF      DF    0092          PUSHAL   DEVDSC                    ; Make sure...
              00A0'CF      DF    0096          PUSHAL   DEVDSC                    ; ...that the specified controller...
   00000000'GF     02      FB    009A          CALLS    #2,G^STR$UPCASE           ; ...is all uppercase for later comaparison
  52   00A0'CF     01      C1    00A1          ADDL3    #1,DEVDSC,R2              ; Est 'ate the eventual...
       00A8'CF     52      A0    00A7          ADDW2    R2,PROCESS_NAME           ; ...p ocess name length (incl. "_")
                           DE    00AC          MOVAL    PROCESS_NAME+8-           ; Locate first available byte...
                                 00AD                   +MAX_PROC_NAME-           ; ...in process name handle...
         50   00B4'CF            00AD                   -PROCESS_NAME_FREE,R0     ; ...for device name
                   0B      C3    00B1          SUBL3    #PROCESS_NAME_FREE,-      ; Will the device name fit...
              51   52            00B3                   R2,R1                     ; ...in the remaining space?
                   08      15    00B5          BLEQ     10$                       ; BR if it will
              50   51      C2    00B7          SUBL2    R1,R0                       ; Overwrite handle otherwise...
       00A8'CF     0F      B0    00BA          MOVW     #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                                 00BF   10$:
         80    5F 8F       90    00BF          MOVB     #^A/_/,(R0)+              ; Separate handle from device name
 60   00BF'CF     00A0'CF  28    00C3          MOVC3    DEVDSC,DEV_NAME,(R0)      ; Concatenate handle with device name
                   7E      D4    00CB          CLRL     -(SP)                     ; Set the time stamp flag
              000F'CF      DF    00CD          PUSHAL   TEST_NAME                 ; Set the test name
                   02      DD    00D1          PUSHL    #2                        ; Push the argument count
        00741039 8F        DD    00D3          PUSHL    #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
   00000000'GF     04      FB    00D9          CALLS    #4,G^LIB$SIGNAL           ; Print the startup message
      000A'CF      20      A8    00E0          BISW2    #BEGIN_MSGM,FLAG          ; Set flag so we don't print it again
                                 00E5          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDR7800_x
                                 00F0
                   02      E1    00F0          BBC      S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
         66 15FC'CF              00F2                   SYSIN_FAB+FAB$L_DEV,20$
                                 00F6          $GETDVI_S DEVNAM = SYS$INPUT,-   ; Get the name of...
                                 00F6                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                 00F6                    ITMLST = INPUT_ITMLST,-
                                 00F6                    IOSB   = QUAD_STATUS
         45 0152'CF        E9    0112          BLBC     QUAD_STATUS,20$           ; Avoid CTRL/C handler if any error
                                 0117          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                 0117                    CHAN   = TTCHAN
                                 0128          $QIOW_S CHAN     = TTCHAN,-        ; Enable CTRL/C AST's...
                                 0128                  FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                                 0128                  P1       = CCASTHAND
              00A8'CF      DF    0149          PUSHAL   PROCESS_NAME              ; ...and tell the user...
                   01      DD    014D          PUSHL    #1                        ; ...
        0074832B 8F        DD    014F          PUSHL    #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
   00000000'GF     03      FB    0155          CALLS    #3,G^LIB$SIGNAL           ; ...
                                 015C   20$:
                                 015C
```

```
                                  015C  ;
                                  015C  ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                  015C  ; and unit configuration and lets us know if the setup to run this test was
                                  015C  ; done correctly.
                                  015C  ;
                                  015C          $OPEN   FAB = INI_FAB,-          ; Open file 'UETINIDEV.DAT"
                                  015C                  ERR = RMS_ERROR
                                  016B          $CONNECT RAB = INI_RAB,-         ; Connect the RAB and FAB
                                  016B                  ERR = RMS_ERROR
                                  017A          $MGBLSC_S  INADR = INADDRESS,-   ; Connect to UETSUPDEV global section
                                  017A                    RETADR = OUTADDRESS,-
                                  017A                    GSDNAM = SUPDEV_GBLSEC,-
                                  017A                     FLAGS = #SEC$M_EXPREG
        00000978 8F    50    D1   0199          CMPL    R0,#SS$_NOSUCHSEC        ; Was the section already there?
                       37    12   01A0          BNEQ    30$                      ; BR if it was...
                                  01A2          $OPEN   FAB = SUP_FAB,-          ; ...else open "UETSUPDEV.DAT"
                                  01A2                  ERR = RMS_ERROR
                                  01B1          $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                  01B1                   INADR = INADDRESS,-
                                  01B1                  RETADR = OUTADDRESS,-
                                  01B1                  GSDNAM = SUPDEV_GBLSEC,-
                                  01B1                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                  01D9  30$:
   56   0166'CF    0162'CF   C3   01D9          SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                  01E1
                                  01E1  FIND_IT:
                                  01E1          $GET    RAB = INI_RAB,-          ; Get the first record
                                  01E1                  ERR = RMS_ERROR
                 0416'CF     DF   01F0          PUSHAL  CONT_DESC                ; Make sure...
                 0416'CF     DF   01F4          PUSHAL  CONT_DESC                ; ...that the controller name...
        00000000'GF    02    FB   01F8          CALLS   #2,G^STR$UPCASE          ; ...is all uppercase letters
           001C'CF  44 8F    91   01FF          CMPB    #^A/D/,BUFFER            ; Is this a DDB?
                       27    13   0205          BEQL    10$                      ; Go on if not
           001C'CF  45 8F    91   0207          CMPB    #^A/E/,BUFFER            ; Is this the end of the file?
                       D2    12   020D          BNEQ    FIND_IT                  ; Continue on if not
                 00A0'CF     DF   020F          PUSHAL  DEVDSC                   ; Push device not supported message
                 00A8'CF     DF   0213          PUSHAL  PROCESS_NAME             ; Parameters on the stack
                       02    DD   0217          PUSHL   #2
             00748333 8F     DD   0219          PUSHL   #UETP$_DENOSU
                       02    F0   021F          INSV    #STS$K_ERROR,-           ; Set the severity code...
                       00         0221                  #STS$V_SEVERITY,-
                   6E  03         0222                  #STS$S_SEVERITY,(SP)
             014E'CF   6E    D0   0224          MOVL    (SP),STATUS              ; ...and save it as the exit status
                       04    DD   0229          PUSHL   #4
                     0A42    31   022B          BRW     ERROR_EXIT               ; Exit in error
                                  022E  10$:
 00BF'CF   0022'CF   016C'CF 29   022E          CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                       A7    12   0238          BNEQ    FIND_IT                  ; BR if not
      16E4'CF   16B0'CF  06  28   023A          MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
           0020'CF  54 8F    91   0242          CMPB    #^A/T/,BUFFER+4          ; Can we test this controller?
                       2F    13   0248          BEQL    FOUND_IT                 ; BR if we can...
                                  024A          $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                                  024A                  OUTLEN = BUFFER_PTR,-
                                  024A                  OUTBUF = FAO_BUF,-
                                  024A                  P1     = #DEVDSC
           014E'CF  14       D0   0263          MOVL    #SS$_BADPARAM,STATUS     ; Set return status
                 0014'CF     DF   0268          PUSHAL  BUFFER_PTR               ; ...
```

```
                  01   DD  026C  1150           PUSHL   #1                          ; ...
         00741132 8F   DD  026E  1151           PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...
                  03   DD  0274  1152           PUSHL   #3                          ; ...
                09F7   31  0276  1153           BRW     ERROR_EXIT                  ; We can't test what we can't test
                           0279  1154
                           0279  1155 FOUND_IT:
                           0279  1156           $GET    RAB = INI_RAB,-             ; Get a record
                           0279  1157                   ERR = RMS_ERROR
             0416'CF   DF  0288  1158           PUSHAL  CONT_DESC                   ; Make sure...
             0416'CF   DF  028C  1159           PUSHAL  CONT_DESC                   ; ...that this line...
      00000000'GF   02 FB  0290  1160           CALLS   #2,G^STR$UPCASE             ; ...is all uppercase letters
       001C'CF   55 8F 91  0297  1161           CMPB    #^A/U/,BUFFER               ; Is this a UCB?
                  24   13  029D  1162           BEQL    30$                         ; BR if it is
       001C'CF   44 8F 91  029F  1163           CMPB    #^A/D/,BUFFER               ; Is this a DDB?
                  19   13  02A5  1164           BEQL    20$                         ; BR if yes
       001C'CF   45 8F 91  02A7  1165           CMPB    #^A/E/,BUFFER               ; Is this the end?
                  11   13  02AD  1166           BEQL    20$                         ; BR if yes
                           02AF  1167 10$:
             0362'CF   DF  02AF  1168           PUSHAL  ILLEGAL_REC                 ; Then this is an error in the record
                  01   DD  02B3  1169           PUSHL   #1                          ; Push the error message
         00741132 8F   DD  02B5  1170           PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
                  03   DD  02BB  1171           PUSHL   #3                          ; Push the temp arg count
                09B0   31  02BD  1172           BRW     ERROR_EXIT                  ; Finish for good
                           02C0  1173 20$:
                0123   31  02C0  1174           BRW     ALL_SET                     ; Found DDB or END
                           02C3  1175 30$:
       0020'CF   54 8F 91  02C3  1176           CMPB    #^A/T/,BUFFER+4             ; Is the unit testable?
                  AE   12  02C9  1177           BNEQ    FOUND_IT                    ; BR if not
                  01   DD  02CB  1178           PUSHL   #1                          ; Flag to ignore blanks when converting
                  02   DD  02CD  1179           PUSHL   #2                          ; Set byte size of results
             016A'CF   DF  02CF  1180           PUSHAL  UNIT_NUMBER                 ; Set address to receive word
             040E'CF   DF  02D3  1181           PUSHAL  UNIT_DESC                   ; Push string address
      00000000'GF   04 FB  02D7  1182           CALLS   #4,G^OTS$CVT_TI_L           ; Convert ASCII unit # to decimal
               CE 50   E9  02DE  1183           BLBC    R0,10$                      ; Don't allow bogus unit to pass
             05   20   3B  02E1  1184           SKPC    #^A/ /,#MAX_UNIT_DESIG,- ; Find out where unit number really is
             0022'CF       02E4  1185                   BUFFER+6
                  50   D7  02E7  1186           DECL    R0                          ; Units must all be at least one digit
         61   50  30   3B  02E9  1187           SKPC    #^A/0/,R0,(R1)              ; Skip leading zeroes on the unit
                  50   D6  02ED  1188           INCL    R0                          ; Compensate for DECL above
   00A0'CF   016C'CF   50  A1  02EF  1189       ADDW3   R0,DEVNAM_LEN,DEVDSC        ; Calculate device'unit string length
         52   016C'CF  3C  02F7  1190           MOVZWL  DEVNAM_LEN,R2               ; Offset to unit number in DEVDSC
    00BF'C2   61   50  28  02FC  1191           MOVC3   R0,(R1),DEV_NAME(R2)        ; Append unit number to device
                           0302  1192           $GETDEV_S DEVNAM = DEVDSC,-         ; Get the device characteristics
                           0302  1193                   PRIBUF = DIB
         57   00DA'CF  9A  0317  1194           MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
         58   00DB'CF  9A  031C  1195           MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
                           0321  1196           $FAO_S  CTRSTR = CS1,-
                           0321  1197                   OUTBUF = FAO_BUF,-
                           0321  1198                   P1     = R7,-
                           0321  1199                   P2     = R8                 ; Make it into a string
0162'DF   56   001C'CF   06  39  0336  1200     MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
                  1E   13  033F  1201           BEQL    40$                         ; BR if it was found
                           0341  1202           $FAO_S  CTRSTR = CS3,-              ; Try for full class support
                           0341  1203                   OUTBUF = FAO_BUF,-
                           0341  1204                   P1 = R7
0162'DF   56   001C'CF   06  39  0354  1205     MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                  0D   12  035D  1206           BNEQ    50$                         ; BR if not found
```

```
                          035F 40$:
      55   000F'CF   9A   035F          MOVZBL  TEST_NAME,R5             ; Get the test name length
0017'CF    63    55   29  0364          CMPC3   R5,(R3),TEST_NAME+8      ; Are we the right test?
                 1F   13  036A          BEQL    60$                      ; BR if yes
                          036C 50$:
           00A0'CF   DF   036C          PUSHAL  DEVDSC                   ; Push device not supported message
           00A8'CF   DF   0370          PUSHAL  PROCESS_NAME             ; Parameters on the stack
                 02   DD  0374          PUSHL   #2                       ; Push the argument count
         00748333 8F  DD  0376          PUSHL   #UETP$_DENOSU
                 02   F0  037C          INSV    #STS$K_ERROR,-
                 00       037E                  #STS$V_SEVERITY,-
           6E    03       037F                  #STS$S_SEVERITY,(SP)     ; Set the severity code...
      014E'CF    6E   D0  0381          MOVL    (SP),STATUS              ; ...and save it as the exit status
                 04   DD  0386          PUSHL   #4                       ; Push the partial arg count...
               08E5   31  0388          BRW     ERROR_EXIT               ; ...and split this scene
```

```
038B  ;+
038B  ; The following code dynamically allocates enough memory for a unit block,
038B  ; a device dependent parameter area and I/O buffers. The unit block is inserted
038B  ; into the queue header UNIT_LIST.  It then initializes the unit block.
038B  ; A comment indicates where the device dependent parameters should be
038B  ; initialized.  The unit block format is as follows:
038B  ;
038B  ;                        +---------------+  --------
038B  ;       UETUNT$L_FLINK   !               !         ^
038B  ;                        +---------------+         !
038B  ;       UETUNT$L_BLINK   !               !         !
038B  ;                        +-----------+---+         !
038B  ;       UETUNT$B_TYPE                !   !         !
038B  ;                            +-------+---+         !
038B  ;       UETUNT$W_SIZE        !   !   !      contains DEVDEP_SIZE + UETUNT$C_INDSIZ
038B  ;                        +---+-------+             !
038B  ;       UETUNT$B_FLAGS   !   !                     !
038B  ;                        +---+   +-------+         !
038B  ;       UETUNT$W_CHAN            !   !   !         !
038B  ;                        +-------+-------+         !
038B  ;       UETUNT$W_FUNC    !   !   !                 !
038B  ;                        +-------+-------+         +----- UETUNT$C_SIZE
038B  ;       UETUNT$L_ITER    !               !         !
038B  ;                        +---------------+         !
038B  ;       UETUNT$T_FILSPC  !               !         !
038B  ;                        !/\/\/\/\/\/\/\/!         !
038B  ;                        NAM$C_MAXRSS bytes        !
038B  ;                        !/\/\/\/\/\/\/\/.         !
038B  ;                        !               !         .
038B  ;       UETUNT$K_FAB     !---------------!         .
038B  ;                        !               .         !
038B  ;                        !/\/\/\/\/\/\/\/.         !
038B  ;                         FAB$C_BLN bytes          .
038B  ;                        !/\/\/\/\/\/\/\/!         !
038B  ;                        !               !         !
038B  ;       UETUNT$K_RAB     +---------------+         !
038B  ;                        !               !         !
038B  ;                        !\/\/\/\/\/\/\/\!         !
038B  ;                         RAB$C_BLN bytes          !
038B  ;                        !\/\/\/\/\/\/\/\!         !
038B  ;                        !               !         v
038B  ;       UETUNT$K_DEVDEP  +---------------+  --------
038B  ;                        !               !         ^
038B  ;                        !\/\/\/\/\/\/\/\!         !
038B  ;                           user defined           +----- DEVDEP_SIZE
038B  ;                        !\/\/\/\/\/\/\/\!         !
038B  ;                        !               !         v
038B  ;                        +---------------+  --------
038B  ;    READ/WRITE buffers  !               !         ^
038B  ;                        !\/\/\/\/\/\/\/\!         !
038B  ;                           user defined           +----- WRITE_SIZE and READ_SIZE
038B  ;                        !/\/\/\/\/\/\/\/!         !
038B  ;                        !               !         v
038B  ;                        +---------------+  --------
038B  ;-
```

```
                            038B  60$:
                            038B          $EXPREG_S PAGCNT = #PAGES,-      ; Get a new node of demand zero memory
                            038B                 RETADR = NEW_NODE
 0198'CF   01A0'DF    5D    039C          INSQTI  @NEW_NODE,UNIT_LIST      ; Put the new node in the unit list
      56   01A0'CF    D0    03A3          MOVL    NEW_NODE,R6              ; Save a copy of its address
      08 A6     01    90    03A8          MOVB    #1,UETUNT$B_TYPE(R6)     ; Set the structure type
         01A4 8F      B0    03AC          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
            09 A6           03B0                  UETUNT$W_SIZE(R6)        ; Set the structure size
   14 A6   00A0'CF    90    03B2          MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
 00A4'DF   00A0'CF    28    03B8          MOVC3   DEVDSC,@DEVDSC+4,-
            15 A6           03BF                  UETUNT$T_FILSPC+1(R6)    ; Save the device name
         0094 8F      28    03C1          MOVC3   #FAB$C_BLN+RAB$C_BLN,-
 0110 C6   173C'CF          03C5                  DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
      57   0110 C6    DE    03CB          MOVAL   UETUNT$K_FAB(R6),R7      ; Save the FAB address
      58   0160 C6    DE    03D0          MOVAL   UETUNT$K_RAB(R6),R8      ; Save the RAB address
     3C A8     57     D0    03D5          MOVL    R7,RAB$L_FAB(R8)         ; Set the FAB address in the RAB
            14 A6     90    03D9          MOVB    UETUNT$T_FILSPC(R6),-
            34 A7           03DC                  FAB$B_FNS(R7)            ; Set the FNS field in the FAB
            15 A6     DE    03DE          MOVAL   UETUNT$T_FILSPC+1(R6),-
            2C A7           03E1                  FAB$L_FNA(R7)            ; Set the FNA field in the FAB
                            03E3  ;
                            03E3  ; Set the device dependent parameters in here
                            03E3  ;
             FE93     31    03E3          BRW     FOUND_IT                 ; Do the next UCB
```

```
                              03E6  ;
                              03E6  ; Arrive here when we have the device configuration.  In normal or loop forever
                              03E6  ; mode, set a timer far enough in the future such that we can do a reasonable
                              03E6  ; set of tests before the timer expires, but if our device gets hung, the
                              03E6  ; program won't waste too much time before noticing.  Let one-shot mode be a
                              03E6  ; special case.
                              03E6  ;
                              03E6 ALL_SET:
              0198'CF   D5    03E6          TSTL    UNIT_LIST                 ; Anything to test?
                   16   12    03EA          BNEQ    10$                       ; BR if yes
              033C'CF   DF    03EC          PUSHAL  NOUNIT_SELECTED           ; Else set up the error message...
                   01   DD    03F0          PUSHL   #1                        ; ...argument count...
          00741132 8F   DD    03F2          PUSHL   #UETP$_TEXT!STS$K_ERROR   ; ...signal name...
                   03   DD    03F8          PUSHL   #3                        ; ...and parameter count
        014E'CF    14   D0    03FA          MOVL    #SS$_BADPARAM,STATUS      ; Set return status
                 086E   31    03FF          BRW     ERROR_EXIT                ; ...and give up, complaining
                              0402 10$:
        000A'CF    04   A8    0402          BISW2   #SAFE_TO_UPDM,FLAG        ; OK safe to update UETINIDEV.DAT now
                              0407          $ASSIGN_S CHAN   = CHAN,-
                              0407                    DEVNAM = DEVDSC         ; Get a channel
                              0418          $LKWSET_S INADR  = CMDBLKDES      ; Lock command block into WS
                              0427                                            ; 11/780 ucode bug in QUEUE
                              0427                                            ; Instructions
    52   00000000'GF    9A    0427          MOVZBL  G^IOC$GW_XFMXRATE,R2      ; Get the current max xfer rate
       12C7'CF     52   90    042E          MOVB    R2,W^CMDTBL+XF$B_CMT_RATE ; Set the data transfer rate
53   00000100 8F   52   C3    0433          SUBL3   R2,#256,R3                ; 256-max xfer rate into R3
           54      53   6E    043B          CVTLD   R3,R4                     ; Convert to double float format
           56      32   50    043E          MOVF    #^F40.,R6                 ; Set the double constant
      57   03C9'CF      DE    0441          MOVAL   DR780,R7                  ; Set default device name
      00DB'CF    00'8F  91    0446          CMPB    #DT$_DR750,DIBBUF+DIB$B_DEVTYPE ; Is it a 750?
                   0C   12    044C          BNEQ    15$                       ; BR if not
    56   00004248 8F    50    044E          MOVF    #^F12.5,R6                ; Set new convertion rate
      57   03CD'CF      DE    0455          MOVAL   DR750,R7                  ; Set correct device name
                              045A 15$:
   0217'CF    56   54   67    045A          DIVD3   R4,R6,W^RATE_FLOAT        ; 40/(256-max xfer rate)
                   03   DD    0460          PUSHL   #3                        ; Push # of digits in the fraction
              0208'CF   DF    0462          PUSHAL  W^RATE_BUF                ; Push string storage desc adr
              0217'CF   7F    0466          PUSHAQ  W^RATE_FLOAT              ; Push adr of floating number
       00000000'GF 03   FB    046A          CALLS   #3,G^FOR$CNV_OUT_F        ; Make the number a string
                              0471          $FAO_S  W^CS2,W^RATE_DESC,W^FAO_BUF,-
                              0471                  R7,R2,#RATE_BUF           ; Make up the message
              021F'CF   DF    048E          PUSHAL  W^RATE_DESC               ; Push the string address
                   01   DD    0492          PUSHL   #1                        ; Push the arg count
          00741133 8F   DD    0494          PUSHL   #UETP$_TEXT!STS$K_INFO    ; Push the signal name
       00000000'GF 03   FB    049A          CALLS   #3,G^LIB$SIGNAL           ; Report the message
                              04A1          $CREMBX_S CHAN   = MBCHAN         ; make a mailbox
                              04B4          $GETCHN_S CHAN   = MBCHAN,-
                              04B4                    PRIBUF = DIB            ; get the unit number of the mailbox
                              04CA          $CREPRC_S IMAGE  = PROCESS,-
                              04CA                    MBXUNT = DIBBUF+DIB$W_UNIT,-
                              04CA                    ERROR  = XFLDR_SYS$ERROR_DESC,-
                              04CA                    PIDADR = PID            ; load the ucode
                              04F4          $WAKE_S PIDADR   = PID            ; wake XFLOADER.EXE from the HIBER
                              0501          $SETIMR_S DAYTIM = ONEMIN,-       ; Catch hungs by ucode loader process
                              0501                    ASTADR = 100$,-
                              0501                    REQIDT = PID
                              0516          $QIOW_S FUNC     = #IO$_READVBLK,-
                              0516                    EFN       = #1,-
                              0516                    CHAN      = MBCHAN,-
                              0516                    P1        = BUFFER,-
                              0516                    P2        = #256          ; read the ucode load results
                              053B                    $CANTIM_S REQIDT = PID    ; Ucode loader process finished
        15 0020'CF     E8     0548                    BLBS      BUFFER+ACC$L_FINALSTS,20$ ; check the load status
           0020'CF     DD     054D                    PUSHL     BUFFER+ACC$L_FINALSTS   ; Push the failure code
           0271'CF     DF     0551                    PUSHAL    ULOAD_FAILED            ; Push failure message address
                01     DD     0555                    PUSHL     #1                      ; Push arg count
       00741132 8F     DD     0557                    PUSHL     #UETP$_TEXT!STS$K_ERROR ; Push the signal name
                04     DD     055D                    PUSHL     #4                      ; Push the temp arg count
              070E     31     055F                    BRW       ERROR_EXIT              ; And die
                              0562       20$:
                              0562                    $OPEN     FAB = XFLDR_SYS$ERROR_FAB ; Did ucode loader find any errors?
50     00000000'8F     D1     056D                    CMPL      #RMS$_FNF,R0            ; If there was no SYS$ERROR file...
                03     12     0574                    BNEQ      30$
              0095     31     0576                    BRW       80$                     ; ...then BR around file copy
                              0579       30$:
             09 50     E8     0579                    BLBS      R0,40$                  ; BR if we can read the file
           17D0'CF     DF     057C                    PUSHAL    XFLDR_SYS$ERROR_FAB     ; If we can't read it...
      0BCA'CF    01    FB     0580                    CALLS     #1,RMS_ERROR            ; ...then complain and bail out
                              0585       40$:
                              0585                    $CONNECT RAB = XFLDR_SYS$ERROR_RAB,-
                              0585                              ERR = RMS_ERROR
                              0594                    $PUTMSG_S MSGVEC = XFLDR_COPY_START ; Announce our intent to copy
                              05A5       50$:
                              05A5                    $GET      RAB = XFLDR_SYS$ERROR_RAB ; Read a line from the file
50     00000000'8F     D1     05B0                    CMPL      #RMS$_EOF,R0            ; Are we beyond the file's end?
                26     13     05B7                    BEQL      70$                     ; BR out of loop if we are
             09 50     E8     05B9                    BLBS      R0,60$                  ; BR if we were able to read a record
           1820'CF     DF     05BC                    PUSHAL    XFLDR_SYS$ERROR_RAB     ; If we were not able to read it...
      0BCA'CF    01    FB     05C0                    CALLS     #1,RMS_ERROR            ; ...then complain and bail out
                              05C5       60$:
           1842'CF     B0     05C5                    MOVW      XFLDR_SYS$ERROR_RAB+-   ; Get the size of this line
           0014'CF            05C9                              RAB$W_RSZ,BUFFER_PTR
                              05CC                    $PUTMSG_S MSGVEC = XFLDR_COPY_LINE ; Report contents of the line
                C6     11     05DD                    BRB       50$                     ; Loop for the next line
                              05DF       70$:
                              05DF                    $PUTMSG_S MSGVEC = XFLDR_COPY_FINISH ; Announce the end of the file
                              05F0                    $CLOSE    FAB = XFLDR_SYS$ERROR_FAB,-
                              05F0                              ERR = RMS_ERROR
                              05FF                    $ERASE    FAB = XFLDR_SYS$ERROR_FAB,-
                              05FF                              ERR = RMS_ERROR
                              060E       80$:
                              060E                    $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                              060E                              RSLLEN = BUFFER_PTR,-
                              060E                              RSLBUF = FAO_BUF
      001C'CF    20    8A     0627                    BICB2     #LC_BITM,BUFFER         ; Convert to upper case
   001C'CF    4F 8F    91     062C                    CMPB      #^A70/,BUFFER           ; Is this a one shot?
                20     12     0632                    BNEQ      TIME_IT                 ; BR if not
      000A'CF    02    A8     0634                    BISW2     #TEST_OVERM,FLAG        ; End after one iteration
                2C     11     0639                    BRB       RESTART                 ; Skip the SETIMR
                              063B
                              063B      100$:         ; Reached by timer AST if microcode loader subprocess fails to
                              063B                    ; return on schedule.  Assume it's hung.
              0000            063B                    .WORD     ^M<>
             04 AC     D5     063D                    TSTL      04(AP)                  ; Was the process even started?
           01  12 0640          BNEQ    110$                        ; BR if it was
               04 0642          RET                                 ; It wasn't - let mainline code handle
                  0643 110$:
      0077'CF  DF 0643          PUSHAL  XFLDR_HUNG                  ; Set up an error message...
           01  DD 0647          PUSHL   #1
  00741132 8F  DD 0649          PUSHL   #UETP$_TEXT!STS$K_ERROR
           03  DD 064F          PUSHL   #3
         061C  31 0651          BRW     ERROR_EXIT                  ; ...and ba l out
                  0654
                  0654 TIME_IT:
                  0654          $SETIMR_S DAYTIM = THREEMIN,-       ; Set timer AST to 3 minutes
                  0654                    ASTADR = TEST_END,-
                  0654                    EFN    = #EFN2
```

```
                               0667              .SBTTL  Test the DR780/DR750
                               0667      RESTART:
                               0667      ;*************************************************************************
                               0667      ;
                               0667      ; Device test specific code goes here.
                               0667      ;
                               0667      ; At this point the device designation is in location DEV_NAME pointed to by
                               0667      ; descriptor DEVDSC. The device is known to be supported and testable by this test.
                               0667      ; To leave successfully BRW SUC_EXIT, to leave in error BRW ERROR_EXIT.
                               0667      ;
                               0667      ;*************************************************************************
                               0667
         000A'CF    10    8A   0667              BICB2   #FPAC_FLGM,FLAG                 ; clear the first packet AST flag
                               066C              $SETIMR_S DAYTIM = TENSEC,-
                               066C                      EFN    = #EFN2,-
                               066C                      ASTADR = HUNG_TEST,-
                               066C                      REQIDT = #1                     ; set 10 sec watch dog timer in case
                               067F                                                      ; the clock jumpers are missing
                               067F              $QIO_S  FUNC   = #IO$_STARTDATA,-       ; Initialize the mapping pointers
                               067F                      CHAN   = CHAN,-                 ; channel must have been assigned.
                               067F                      EFN    = #EFN1,-                ; EF
                               067F                      IOSB   = DRIOSTAT,-             ; IO status block.
                               067F                      ASTADR = IO_COMPLETE,-          ; ast to be taken when dr halts
                               067F                      P1     = CMDTBL,-               ; address of the command table
                               067F                      P2     = #XF$K_CMT_LENGTH       ; length of command table.
         000A'CF    09    93   06A6              BITB    #CC_FLGM!ERR_FLGM,FLAG          ; control C or error occured?
                    03    13   06AB              BEQL    5$                              ; br if no
                  061D    31   06AD              BRW     ERROR_EXIT1                     ; br if yes
                               06B0      5$:
         5A   056C'CF     DE   06B0              MOVAL   PKT_TBL,R10                     ; set pkt address pointer
         5B   12D8'CF     DE   06B5              MOVAL   INPTQH,R11                      ; set the queue header pointer
               59   6A    D0   06BA              MOVL    (R10),R9                        ; get the packet address
               1C   A9    D4   06BD              CLRL    XF$L_PKT_DSL(R9)                ; init the DSL
                               06C0              QRETRY  ERROR = BADQUEUE,-
                               06C0              INSQTI  @(R10)+,FREEQH                  ; put a free packet on the free queu
                               06D0              .REPT   PKT_COUNT-1
                               06D0              MOVL    (R10),R9                        ; get the packet address
                               06D0              CLRL    XF$L_PKT_DSL(R9)                ; init the DSL
                               06D0              QRETRY  ERROR = BADQUEUE,-
                               06D0              INSQTI  @(R10)+,(R11)                   ; put a command packet on the input
               59   6A    D0   06D0              .ENDR
         12CF'DF    01    D0   07C0              MOVL    #1,@GOBIT                       ; give it heck!
                               07C5              $WAITFR_S EFN = #EFN1                   ; wait for further AST's or iteratio
         000A'CF    09    93   07CE              BITB    #CC_FLGM!ERR_FLGM,FLAG          ; control C or error occured?
                    03    13   07D3              BEQL    10$                             ; br if no
                  04F5    31   07D5              BRW     ERROR_EXIT1                     ; br if yes
                               07D8      10$:
              0176'CF     D6   07D8              INCL    ITERATION                       ; increment iteration count
01AC'CF  00006030 8F      D0   07DC              MOVL    #UNUSED_FUNC,PACK_REMOVED       ; mask out unused packet types
         0A46'CF    00    FB   07E5              CALLS   #0,PKT_CHECK                    ; check the packet status and count
         000A'CF    08    93   07EA              BITB    #ERR_FLGM,FLAG                  ; were any DSL's bad?
                    03    13   07EF              BEQL    20$                             ; br if not
                  04D9    31   07F1              BRW     ERROR_EXIT1                     ; fatal error
                               07F4      20$:
         01A8'CF    0D    91   07F4              CMPB    #PKT_COUNT,PKT_CNT              ; right number of packets?
                    69    13   07F9              BEQL    50$                             ; br if yes
                    55    D4   07FB              CLRL    R5                              ; set starting position
```

```
                58   11    D0  07FD  1489         MOVL    #NO_OF_POS_PKTS,R8                 ; set starting size
          52  01B0'CF      DE  0800  1490         MOVAL   ARGS,R2                            ; set arg pointer
                82   0D    D0  0805  1491         MOVL    #PKT_COUNT,(R2)+                   ; set expected pkt count
          82  01A8'CF      D0  0808  1492         MOVL    PKT_CNT,(R2)+                      ; set received pkt count
     82   0D  01A8'CF      C3  080D  1493         SUBL3   PKT_CNT,#PKT_COUNT,(R2)+           ; set the argument count
                                0813  1494 30$:
 56   01AC'CF   58   55    EB  0813  1495         FFC     R5,R8,PACK_REMOVED,R6              ; find a missing packet
                     17    13  081A  1496         BEQL    40$
          58   11    56    C3  081C  1497         SUBL3   R6,#NO_OF_POS_PKTS,R8              ; update the size
          55   01    56    C1  0820  1498         ADDL3   R6,#1,R5                           ; update the starting position
               56    08    C4  0824  1499         MULL2   #8,R6                              ; make it a byte displacement
        56   000004E4'8F   C0  0827  1500         ADDL2   #NAME_TBL,R6                       ; make it an address
               82    66    D0  082E  1501         MOVL    (R6),(R2)+                         ; save it in the argument list
                     E0    11  0831  1502         BRB     30$                                ; get the next packet that is missin
                                0833  1503 40$:
                                0833  1504         $FAOL_S CTRSTR = CS4,-
                                0833  1505                 OUTLEN = BADRPKT,-
                                0833  1506                 OUTBUF = FAO_BUF,-
                                0833  1507                 PRMLST = ARGS                      ; convert into a string
          007410E2 8F      D0  084A  1508         MOVL    #UETP$_ABENDD!STS$K_ERROR,-
               014E'CF         0850  1509                 STATUS                             ; set an exit status
               0200'CF     DF  0853  1510         PUSHAL  BADRPKT                            ; push the constructed message
                    01     DD  0857  1511         PUSHL   #1                                 ; push arg count
          00741132 8F      DD  0859  1512         PUSHL   #UETP$_TEXT!STS$K_ERROR            ; push the signal name
                    03     DD  085F  1513         PUSHL   #3                                 ; Push temp arg count
                  040C     31  0861  1514         BRW     ERROR_EXIT                         ; bail out
                                0864  1515 50$:
      1591'CF   1591'CF    92  0864  1516         MCOMB   DIAG_CNTRL_MESS,DIAG_CNTRL_MESS ; toggle the control message
         000A'CF    02     93  086B  1517         BITB    #TEST_OVERM,FLAG                   ; is the test over?
                    03     13  0870  1518         BEQL    60$                                ; br if no
                  0115     31  0872  1519         BRW     SUC_EXIT                           ; br if yes
                                0875  1520 60$:
                  FDEF     31  0875  1521         BRW     RESTART                            ; do it again!
                                0878  1522 ;
                                0878  1523 ; Packet AST routine
                                0878  1524 ;
                                0878  1525 PKT1_AST:
                          0FFC  0878  1526         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      54 000A'CF    04     E2  087A  1527         BBSS    #FPAC_FLGV,FLAG,10$                ; only check data on the first packe
 1233'CF   0233'CF  007C 8F  29  0880  1528       CMPC3   #124,OUTPUT_BUF+4,INPUT1_BUF+4     ; check the data
                    48     13  088A  1529         BEQL    10$                                ; br if OK
               014A'CF     D6  088C  1530         INCL    ERROR_COUNT                        ; bump the cum. error cnt
               7E   63     9A  0890  1531         MOVZBL  (R3),-(SP)                         ; get bad byte
               7E   61     9A  0893  1532         MOVZBL  (R1),-(SP)                         ; get good byte
   7E     0000007C 8F  50  C3  0896  1533         SUBL3   R0,#124,-(SP)                      ; get the byte number
                    00     DD  089E  1534         PUSHL   #0                                 ; push the unit number
               00A0'CF     DF  08A0  1535         PUSHAL  DEVDSC                             ; push the controller name
          000F0005 8F      DD  08A4  1536         PUSHL   #^XF0005                           ; push arg count
          00748012 8F      DD  08AA  1537         PUSHL   #UETP$_DATAER!STS$K_ERROR          ; push the signal name
               014A'CF     DD  08B0  1538         PUSHL   ERROR_COUNT                        ; push cumulative error count
               00A8'CF     DF  08B4  1539         PUSHAL  PROCESS_NAME
          00010002 8F      DD  08B8  1540         PUSHL   #^X10002                           ; push arg count
          00748022 8F      DD  08BE  1541         PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR       ; push the signal name
       00000000'GF   0B    FB  08C4  1542         CALLS   #11,G^LIB$SIGNAL                   ; output the message
          00748012 8F      D0  08CB  1543         MOVL    #UETP$_DATAER!STS$K_ERROR,-
               014E'CF         08D1  1544                 STATUS                             ; push the signal name
                                08D4  1545 10$:
```

```
 007C 8F    00   122F'CF   00   2C  08D4          MOVC5   #0,INPUT1_BUF,#0,#124,INPUT1_BUF+4 ; clear the data buffer
                  1233'CF           08DD
                                04  08E0          RET
                                    08E1   ;
                                    08E1   ; Watch dog timer will come to here if the DR does not complete one
                                    08E1   ; execution of all packets within 10 seconds.
                                    08E1   ;
                                    08E1   HUNG_TEST:
                              0FFC  08E1          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; entry mask
             000A'CF    08      88  08E3          BISB2   #ERR_FLGM,FLAG                      ; set the error flag
     014E'CF   0000022C 8F      D0  08E8          MOVL    #SS$_TIMEOUT,STATUS                 ; Set the exit status
                  014A'CF       D6  08F1          INCL    ERROR_COUNT                         ; bump the error counter
                  04A0'CF       DF  08F5          PUSHAL  TEST_HUNG                           ; push the constructed message
               000F0001 8F      DD  08F9          PUSHL   #^XF0001                            ; push arg count
               00741132 8F      DD  08FF          PUSHL   #UETP$_TEXT!STS$K_ERROR             ; push the signal name
                  014A'CF       DD  0905          PUSHL   ERROR_COUNT                         ; push cumulative error count
                  00A8'CF       DF  0909          PUSHAL  PROCESS_NAME
               00010002 8F      DD  090D          PUSHL   #^X10002                            ; push the arg count
               00748022 8F      DD  0913          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR        ; push the signal name
          00000000'GF   07      FB  0919          CALLS   #7,G^LIB$SIGNAL                     ; output the message
                                    0920          $SETEF_S EFN=#EFN1                          ; time to wake up
                                04  0929          RET                                         ; go and fail
                                    092A   IO_COMPLETE:
                              0000  092A          .WORD   0                                   ; QIO completion ast entry point
                                    092C          $CANTIM_S REQIDT=#1                         ; stop the watch dog timer only
           01   0227'CF         B1  0937          CMPW    DRIOSTAT,#SS$_NORMAL                ; check the IO status
                       3F       13  093C          BEQL    10$                                 ; br if OK
       014E'CF   0227'CF        B0  093E          MOVW    DRIOSTAT,STATUS                     ; save the status
                  014A'CF       D6  0945          INCL    ERROR_COUNT                         ; bump the error count
                  0227'CF       DD  0949          PUSHL   DRIOSTAT                            ; push the DSL
                  028F'CF       DF  094D          PUSHAL  START_DATA_FAILED
               000F0001 8F      DD  0951          PUSHL   #^XF0001
               00741132 8F      DD  0957          PUSHL   #UETP$_TEXT!STS$K_ERROR
                  014A'CF       DD  095D          PUSHL   ERROR_COUNT                         ; push cumulative error count
                  00A8'CF       DF  0961          PUSHAL  PROCESS_NAME
               00010002 8F      DD  0965          PUSHL   #^X10002                            ; push arg count
               00748022 8F      DD  096B          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR        ; push the signal name
          00000000'GF   08      FB  0971          CALLS   #8,G^LIB$SIGNAL                     ; output the message
             000A'CF    08      88  0978          BISB2   #ERR_FLGM,FLAG                      ; set error flag
                                    097D   10$:
                  0227'CF       D4  097D          CLRL    DRIOSTAT                            ; reset the DR's IO status block
                                04  0981          RET                                         ; and return
                                    0982   TEST_END:
                              0000  0982          .WORD   0                                   ; entry mask
             000A'CF    02      88  0984          BISB2   #TEST_OVERM,FLAG                    ; set the test ended flag
                                04  0989          RET                                         ; return
                                    098A   ;
                              098A  1593
                              098A  1594 SUC_EXIT:
                              098A  1595        $TRNLOG_S LOGNAM = MODE,-
                              098A  1596                  RSLLEN = BUFFER_PTR,-
                              098A  1597                  RSLBUF = FAO_BUF        ; Get the run mode
         001C'CF   20    8A   09A3  1598        BICB2   #LC_BITM,BUFFER           ; Convert to upper case
     001C'CF    4C 8F    91   09A8  1599        CMPB    #^A?L/,BUFFER             ; Is this a loop for ever?
                   40    12   09AE  1600        BNEQ    10$                       ; BR if not
         000A'CF   02    AA   09B0  1601        BICW2   #TEST_OVERM,FLAG          ; Reset the termination flag
              017A'CF    D6   09B5  1602        INCL    PASS                      ; Bump the pass count
                              09B9  1603        $FAO_S  CTRSTR = PASS_MSG,-
                              09B9  1604                OUTLEN = BUFFER_PTR,-
                              09B9  1605                OUTBUF = FAO_BUF,-
                              09B9  1606                P1     = PASS,-
                              09B9  1607                P2     = ITERATION,-
                              09B9  1608                P3     = #0               ; Make the end of pass message
              0014'CF    DF   09D6  1609        PUSHAL  BUFFER_PTR                ; Push the string desc.
                   01    DD   09DA  1610        PUSHL   #1                        ; Push arg count
          00741133 8F    DD   09DC  1611        PUSHL   #UETP$_TEXT!STS$K_INFO    ; Push the signal name
      00000000'GF   03    FB   09E2  1612        CALLS   #3,G^LIB$SIGNAL           ; Print the end of pass message
              0176'CF    D4   09E9  1613        CLRL    ITERATION                 ; Reset the iteration count
                 FC64    31   09ED  1614        BRW     TIME_IT                   ; Do the next pass
                              09F0  1615 10$:
56   0198'CF   00000198'8F    C1   09F0  1616        ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                   02    88   09FA  1617        BISB2   #UETUNT$M_TESTABLE,-
                0B A6         09FC  1618                UETUNT$B_FLAGS(R6)        ; Set the testable bit
     014E'CF   10000001 8F    D0   09FE  1619        MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                              0A07  1620        $EXIT_S STATUS                    ; Exit with the status
```

```
                          0A12          .SBTTL  RANBUF
                          0A12   ;++
                          0A12   ; FUNCTIONAL DESCRIPTION:
                          0A12   ;       This routine fills buffer TEST_DATA with random numbers.
                          0A12   ;
                          0A12   ; CALLING SEQUENCE:
                          0A12   ;       BSBW    RANBUF
                          0A12   ;
                          0A12   ; INPUT PARAMETERS:
                          0A12   ;       NONE
                          0A12   ;
                          0A12   ; OUTPUT PARAMETERS:
                          0A12   ;       BUFSIZ bytes of random data are left in buffer TEST_DATA
                          0A12   ;
                          0A12   ;--
                          0A12
                          0A12   RANBUF:
      52   022F'CF    DE  0A12          MOVAL   TEST_DATA,R2                    ; set buffer adr
      53   0200 8F    3C  0A17          MOVZWL  #BUFSIZ/4,R3                    ; BUFSIZ/4 bytes is the size
                          0A1C   10$:
 016E'CF   0172'CF    C0  0A1C          ADDL2   RANDOM2,RANDOM1                 ; make a new random number
      82   016E'CF    D0  0A23          MOVL    RANDOM1,(R2)+                   ; put it in the buffer
              F1 53   F5  0A28          SOBGTR  R3,10$                          ; do the whole thing!
                      05  0A2B          RSB                                     ; return
                      0A2C          .SBTTL  BADQUEUE
                      0A2C      ;++
                      0A2C      ; FUNCTIONAL DESCRIPTION:
                      0A2C      ;       This routine indicates a bad queue entry has been discovered by the
                      0A2C      ;       QRETRY macro and an error is reported.
                      0A2C      ;
                      0A2C      ; CALLING SEQUENCE:
                      0A2C      ;       BRW     BADQUEUE
                      0A2C      ;
                      0A2C      ; INPUT PARAMETERS:
                      0A2C      ;       ERROR_COUNT = current cumulative error count
                      0A2C      ;
                      0A2C      ; OUTPUT PARAMETERS:
                      0A2C      ;       ERROR_COUNT = bumped by one
                      0A2C      ;
                      0A2C      ;--
                      0A2C
                      0A2C    BADQUEUE:
007410E2 8F     D0    0A2C            MOVL    #UETP$_ABENDD!STS$K_ERROR,-
    014E'CF           0A32                    STATUS                          ; set the status code
    0472'CF     DF    0A35            PUSHAL  BADQUE                          ; push message address
         01     DD    0A39            PUSHL   #1                              ; push arg count
00741132 8F     DD    0A3B            PUSHL   #UETP$_TEXT!STS$K_ERROR         ; push signal name
         03     DD    0A41            PUSHL   #3                              ; Push temp arg count
       022A     31    0A43            BRW     ERROR_EXIT
```

```
                                0A46  ;
                                0A46          .SBTTL  PKT_CHECK
                                0A46  ;++
                                0A46  ; FUNCTIONAL DESCRIPTION:
                                0A46  ;       Routine to check DR packet status off the termination queue.
                                0A46  ;       Each packet is removed from the termination queue and the
                                0A46  ;       DSL is checked for XF$M_IOS_SUCCES and XF$M_IOS_CMDSTD. A
                                0A46  ;       total count is maintianed for each call to the routine at
                                0A46  ;       location PKT_CNT.
                                0A46  ;
                                0A46  ; CALLING SEQUENCE:
                                0A46  ;       CALLS   #0,PKT_CHECK
                                0A46  ;
                                0A46  ; INPUT PARAMETERS:
                                0A46  ;       TERMQH = termination queue head
                                0A46  ;
                                0A46  ; OUTPUT PARAMETERS:
                                0A46  ;       PKT_CNT = number of packets serviced for this call
                                0A46  ;       FLAG  = BIT1 set if an error is encountered
                                0A46  ;       PACK_REMOVED =  bit mask record of which packets were removed
                                0A46  ;                       from the termination queue
                                0A46  ;
                                0A46  ;--
                                0A46  PKT_CHECK:
                        0FFC    0A46          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
             01A8'CF      94    0A48          CLRB    PKT_CNT                         ; set packet count to zero
                                0A4C  PC1:
                                0A4C          QRETRY  ERROR = BADQUEUE,-
                                0A4C          REMQHI  TERMQH,R2                       ; get a packet from the queue
                  01      1C    0A5C          BVC     5$                              ; br if an entry removed
                          04    0A5E          RET                                     ; return
                                0A5F  5$:
             01A8'CF      96    0A5F          INCB    PKT_CNT                         ; bump the packet count
          1C A2    08     D3    0A63          BITL    #XF$M_PKT_FREQPK,XF$L_PKT_DSL(R2) ; is it a free queue packet?
                   05     13    0A67          BEQL    10$                             ; br if not
             53    10     D0    0A69          MOVL    #16,R3                          ; set the free packet index
                   06     11    0A6C          BRB     15$                             ; carry on
                                0A6E  10$:
             04    00     EF    0A6E          EXTZV   #XF$V_PKT_FUNC,#XF$S_PKT_FUNC,-
          53    0A A2           0A71                  XF$B_PKT_CMDCTL(R2),R3  ; get the packet command
                                0A74  15$:
   01AC'CF    01    53    01    F0    0A74          INSV    #1,R3,#1,PACK_REMOVED   ; record it's removal
                    53    08    C4    0A7B          MULL2   #8,R3                   ; make it an index
        53   000004E4'8F    C0    0A7E          ADDL2   #NAME_TBL,R3                    ; make it an ascic packet type pointer
          54    53    04    C1    0A85          ADDL3   #4,R3,R4                        ; make the status address
          1C A2    64     D1    0A89          CMPL    (R4),XF$L_PKT_DSL(R2)           ; is the DSL correct?
                   BD     13    0A8D          BEQL    PC1                             ; br if OK
                                0A8F  20$:
         007410E2 8F      D0    0A8F          MOVL    #UETP$_ABENDD!STS$K_ERROR,-
             014E'CF            0A95                  STATUS                          ; set the status code
             014A'CF      D6    0A98          INCL    ERROR_COUNT                     ; bump the error counter
                                0A9C          $FAO_S  CTRSTR = CS,-
                                0A9C                  OUTLEN = BADRPKT,-
                                0A9C                  OUTBUF = FAO_BUF,-
                                0A9C                  P1     = (R3),-
                                0A9C                  P2     = XF$L_PKT_DSL(R2),-
                                0A9C                  P3     = (R4)           ; create the error string
         0200'CF   DF  ^AB6          PUSHAL  BADRPKT                    ; set message address
      000F0001 8F  DD  0ABA          PUSHL   #^XF0001                   ; set arg count
      00741132 8F  DD  0AC0          PUSHL   #UETP$_TEXT!STS$K_ERROR ; set signal name
         014A'CF   DD  0AC6          PUSHL   ERROR_COUNT                ; push cumulative error count
         00A8'CF   DF  0ACA          PUSHAL  PROCESS_NAME
      00010002 8F  DD  0ACE          PUSHL   #^X10002                   ; push arg count
      00748022 8F  DD  0AD4          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; set signal name
   00000000'GF  07  FB  0ADA          CALLS   #7,G^LIB$SIGNAL            ; output the message
      000A'CF   08  88  0AE1          BISB2   #ERR_FLGM,FLAG             ; set the error flag
                    04  0AE6          RET                                ; return
```

```
0AE7          .SBTTL  System Service Exception Handler
0AE7    ;++
0AE7    ; FUNCTIONAL DESCRIPTION:
0AE7    ;       This routine is executed if a software or hardware exception occurs or
0AE7    ;       if a LIB$SIGNAL system service is used to output a message.
0AE7    ;
0AE7    ; CALLING SEQUENCE:
0AE7    ;       Entered via an exception from the system
0AE7    ;
0AE7    ; INPUT PARAMETERS:
0AE7    ;       ERROR_COUNT   = previous cumulative error count
0AE7    ;                       -----------------
0AE7    ;          AP ---->     !       2       !
0AE7    ;                       !---------------!
0AE7    ;                       ! SIGNL ARY PNT !
0AE7    ;                       !---------------!
0AE7    ;                       ! MECH  ARY PNT !
0AE7    ;                       !---------------!  ---------
0AE7    ;                       !       4       !          ^
0AE7    ;                       !---------------!          !
0AE7    ;                       ! ESTABLISH FP  !          !
0AE7    ;                       !---------------!          !
0AE7    ;                       !     DEPTH     !  Mechanism Array
0AE7    ;                       !---------------!          !
0AE7    ;                       !      R0       !          !
0AE7    ;                       !---------------!          !
0AE7    ;                       !      R1       !          v
0AE7    ;                       !---------------!  ---------
0AE7    ;                       !       N       !          ^
0AE7    ;                       !---------------!          !
0AE7    ;                       ! CONDITION NAME!          !
0AE7    ;                       !---------------!          !
0AE7    ;                       ! N-3 ADDITIONAL!     Signal Array
0AE7    ;                       ! LONG WORD ARGS!          !
0AE7    ;                       !---------------!          !
0AE7    ;                       !      PC       !          !
0AE7    ;                       !---------------!          !
0AE7    ;                       !      PSL      !          v
0AE7    ;                       !---------------!  ---------
0AE7    ; IMPLICIT INPUTS:
0AE7    ;       NONE
0AE7    ;
0AE7    ; OUTPUT PARAMETERS:
0AE7    ;       NONE
0AE7    ;
0AE7    ; IMPLICIT OUTPUTS:
0AE7    ;       NONE
0AE7    ;
0AE7    ; COMPLETION CODES:
0AE7    ;       SS$_NORMAL if it's a UETP condition or RMS error.
0AE7    ;       Error status from exception, otherwise.
0AE7    ;
0AE7    ; SIDE EFFECTS:
0AE7    ;       May branch to ERROR_EXIT.
0AE7    ;       May print a message.
0AE7    ;--
0AE7
```

```
                                0AE7  SSERROR:
                        0FFC    0AE7            .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0AE9
                                0AE9            $SETAST_S ENBFLG = #0           ; Disable AST delivery
                  01      DD    0AF2            PUSHL   #1                      ; Assume ASTs were enabled
            50    09      D1    0AF4            CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                  02      13    0AF7            BEQL    10$                     ; BR if they were
                  6E      D4    0AF9            CLRL    (SP)                    ; Set ASTs to remain disabled
                                0AFB  10$:
                                0AFB            $SETSFM_S ENBFLG = #0           ; Disable SS failure mode
                  01      DD    0B04            PUSHL   #1                      ; Assume SS failure mode was enabled
            50    09      D1    0B06            CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
                  02      13    0B09            BEQL    20$                     ; BR if it was
                  6E      D4    0B0B            CLRL    (SP)                    ; Set SS failure mode to remain off
                                0B0D  20$:
       56      04 AC      D0    0B0D            MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
       59      04 A6      7D    0B11            MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                  10      ED    0B15            CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                  0C            0B17                    #STS$S_FAC_NO,-
  00000074 8F     59            0B18                    R9,#UETP$_FACILITY
                  14      12    0B1E            BNEQ    30$                     ; BR if this is not a UETP exception
            66    02      C2    0B20            SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                                0B23            $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                  21      11    0B32            BRB     40$                     ; Restore ASTs and SS fail mode
                                0B34  30$:
  59    0000045C 8F       D1    0B34            CMPL    #SS$_SSFAIL,R9          ; RMS failures are SysSvc failures
                  32      12    0B3B            BNEQ    50$                     ; BR if this can't be an RMS failure
                  10      ED    0B3D            CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                  0C            0B3F                    #STS$S_FAC_NO,-
            01    5A            0B40                    R10,#RMS$_FACILITY
                  2B      12    0B42            BNEQ    50$                     ; BR if not
  5A    F0000000 8F       CA    0B44            BICL2   #^XF0000000,R10         ; Strip control bits from status code
        08 A6     04      39    0B4B            MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                  14            0B4F                    #NRAT_LENGTH,-
           0170'CF              0B50                    NO_RMS_AST_TABLE        ; ...no AST can be delivered?
                  1A      13    0B53            BEQL    50$                     ; BR if so - must give error here
                                0B55  40$:
                  01      BA    0B55            POPR    #^M<R0>                 ; Restore SS failure mode...
                                0B57            $SETSFM_S ENBFLG = R0           ; ...
                  01      BA    0B60            POPR    #^M<R0>                 ; Restore AST enable...
                                0B62            $SETAST_S ENBFLG = R0           ; ...
            50    01      D0    0B6B            MOVL    S^#SS$_NORMAL,R0        ; Supply a standard status for exit
                          04    0B6E            RET                             ; Resume processing (or goto RMS_ERROR)
                                0B6F  50$:
        014E'CF   59      D0    0B6F            MOVL    R9,STATUS               ; Save the status
                  58      D4    0B74            CLRL    R8                      ; Assume for now it's not SS failure
  59    0000045C 8F       D1    0B76            CMPL    #SS$_SSFAIL,R9          ; But is it a System Service failure?
                  38      12    0B7D            BNEQ    70$                     ; BR if not - no special case message
                                0B7F            $GETMSG_S MSGID = R10,-         ; Get SS failure code associated text
                                0B7F                     MSGLEN = BUFFER_PTR,-
                                0B7F                     BUFADR = FAO_BUF,-
                                0B7F                     FLAGS  = #14,-
                                0B7F                     OUTADR = MSG_BLOCK
           017F'CF        95    0B96            TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                  16      13    0B9A            BEQL    60$                     ; Don't use $GETMSG if no $FAO args...
           0014'CF        DF    0B9C            PUSHAL  BUFFER_PTR              ; ...else build up...
                  01      DD    0BA0            PUSHL   #1                      ; ...a message describing...
        00741130 8F    DD  0BA2  1855            PUSHL   #UETP$_TEXT               ; ...why the System Service failed
              00    5A  F0  0BA8  1856            INSV    R10,#STS$V_SEVERITY,-    ; Give the message...
              6E    03      0BAB  1857                    #STS$S_SEVERITY,(SP)     ; ...the correct severity code
              58    03  D0  0BAD  1858            MOVL    #3,R8                    ; Count the number of args we pushed
                    05  11  0BB0  1859            BRB     70$
                            0BB2  1860 60$:
                    5A  DD  0BB2  1861            PUSHL   R10                      ; Save SS failure code
              58    01  D0  0BB4  1862            MOVL    #1,R8                    ; Count the number of args we pushed
                            0BB7  1863 70$:
         57   66    04  C5  0BB7  1864            MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
              5E    57  C2  0BBB  1865            SUBL2   R7,SP                      ; Save the current signal array...
      6E   04 A6    57  28  0BBE  1866            MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
         7E   66    58  C1  0BC3  1867            ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                  00A6  31  0BC7  1868            BRW     ERROR_EXIT
```

```
                              0BCA          .SBTTL  RMS Error Handler
                              0BCA     ;++
                              0BCA     ; FUNCTIONAL DESCRIPTION:
                              0BCA     ;        This routine handles error returns from RMS calls.
                              0BCA     ;
                              0BCA     ; CALLING SEQUENCE:
                              0BCA     ;        Called by RMS when a file processing error is found.
                              0BCA     ;
                              0BCA     ; INPUT PARAMETERS:
                              0BCA     ;        The FAB or RAB associated with the RMS call.
                              0BCA     ;
                              0BCA     ; IMPLICIT INPUTS:
                              0BCA     ;        NONE
                              0BCA     ;
                              0BCA     ; OUTPUT PARAMETERS:
                              0BCA     ;        NONE
                              0BCA     ;
                              0BCA     ; IMPLICIT OUTPUTS:
                              0BCA     ;        Error message
                              0BCA     ;
                              0BCA     ; COMPLETION CODES:
                              0BCA     ;        NONE
                              0BCA     ;
                              0BCA     ; SIDE EFFECTS:
                              0BCA     ;        Program may exit, depending on severity of the error.
                              0BCA     ;
                              0BCA     ;--
                              0BCA
                              0BCA     RMS_ERROR:
                        0FFC  0BCA             .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              0BCC
          56    04 AC    D0   0BCC             MOVL    4(AP),R6                     ; See whether we're dealing with...
             66    03    91   0BD0             CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                   16    12   0BD3             BNEQ    10$                          ; BR if it's a RAB
         57   041E'CF    DE   0BD5             MOVAL   FILE,R7                      ; FAB-specific code:  text string...
             58    56    D0   0BDA             MOVL    R6,R8                        ; ...address of FAB...
                0C A6    DD   0BDD             PUSHL   FAB$L_STV(R6)                ; ...STV field for error...
                08 A6    DD   0BE0             PUSHL   FAB$L_STS(R6)                ; ...STS field for error...
      014E'CF   08 A6    D0   0BE3             MOVL    FAB$L_STS(R6),STATUS         ; ...and save the error code
                   15    11   0BE9             BRB     COMMON                       ; FAB and RAB share other code
                              0BEB     10$:
         57   042A'CF    DE   0BEB             MOVAL   RECORD,R7                    ; RAB-specific code:  text string...
          58    3C A6    D0   0BF0             MOVL    RAB$L_FAB(R6),R8             ; ...address of associated FAB...
                0C A6    DD   0BF4             PUSHL   RAB$L_STV(R6)                ; ...STV field for error...
                08 A6    DD   0BF7             PUSHL   RAB$L_STS(R6)                ; ...STS field for error...
      014E'CF   08 A6    D0   0BFA             MOVL    RAB$L_STS(R6),STATUS         ; ...and save the error code
                              0C00     COMMON:
          5A    34 A8    9A   0C00             MOVZBL  FAB$B_FNS(R8),R10            ; Get the file name size
                              0C04             $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                              0C04                     OUTLEN = BUFFER_PTR,-
                              0C04                     OUTBUF = FAO_BUF,-
                              0C04                     P1     = R7,-
                              0C04                     P2     = R10,-
                              0C04                     P3     = FAB$L_FNA(R8)
              0014'CF    DF   0C1E             PUSHAL  BUFFER_PTR                   ; ...and arguments for ERROR_EXIT...
                   01    DD   0C22             PUSHL   #1                           ; ...
       00741130 8F       DD   0C24             PUSHL   #UETP$_TEXT                  ; ...
```

```
            00   EF  0C2A  1927        EXTZV   #STS$V_SEVERITY,-
            03       0C2C  1928                #STS$S_SEVERITY,-
      59  014E'CF    0C2D  1929                STATUS,R9              ; ...get the severity code...
         6E   59   88  0C31  1930        BISB2   R9,(SP)                ; ...and add it into the signal name
            05   DD  0C34  1931        PUSHL   #5                     ; Current arg count
          0037   31  0C36  1932        BRW     ERROR_EXIT
```

```
                                  0C39          .SBTTL  CTRL/C Handler
                                  0C39  ;++
                                  0C39  ; FUNCTIONAL DESCRIPTION:
                                  0C39  ;       This routine handles CTRL/C AST's
                                  0C39  ;
                                  0C39  ; CALLING SEQUENCE:
                                  0C39  ;       Called via AST
                                  0C39  ;
                                  0C39  ; INPUT PARAMETERS:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ; IMPLICIT INPUTS:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ; OUTPUT PARAMETERS:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ; IMPLICIT OUTPUTS:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ; COMPLETION CODES:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ; SIDE EFFECTS:
                                  0C39  ;       NONE
                                  0C39  ;
                                  0C39  ;--
                                  0C39
                                  0C39  CCASTHAND:
                            0FFC  0C39          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                  0C3B
             02B4'CF     DF       0C3B          PUSHAL  CNTRLCMSG                   ; Set message pointer
                  01     DD       0C3F          PUSHL   #1                          ; Set arg count
         00741130 8F     DD       0C41          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                  00     DD       0C47          PUSHL   #0                          ; Indicate an abnormal termination
             00A8'CF     DF       0C49          PUSHAL  PROCESS_NAME                ; ...
                  02     DD       0C4D          PUSHL   #2                          ; ...
         007410E0 8F     DD       0C4F          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
     00000000'GF    07   FB       0C55          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                         D0       0C5C          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                                  0C5D                  SS$_CONTROLC--
                                  0C5D                  STS$K_SUCCESS+STS$K_WARNING>,-
014E'CF   10000650 8F             0C5D                  STATUS
                                  0C65          $EXIT_S STATUS                      ; Terminate program cleanly
```

```
                                0C70        .SBTTL  Error Exit
                                0C70  ;++
                                0C70  ; FUNCTIONAL DESCRIPTION:
                                0C70  ;       This routine prints an error message and exits.
                                0C70  ;
                                0C70  ; CALLING SEQUENCE:
                                0C70  ;       MOVx  error status value,STATUS
                                0C70  ;       PUSHx error specific information on the stack
                                0C70  ;       PUSHL current argument count
                                0C70  ;       BRW   ERROR_EXIT
                                0C70  ;
                                0C70  ; INPUT PARAMETERS:
                                0C70  ;       Arguments to LIB$SIGNAL, as above
                                0C70  ;
                                0C70  ; IMPLICIT INPUTS:
                                0C70  ;       NONE
                                0C70  ;
                                0C70  ; OUTPUT PARAMETERS:
                                0C70  ;       Message to SYS$OUTPUT and SYS$ERROR
                                0C70  ;
                                0C70  ; IMPLICIT OUTPUTS:
                                0C70  ;       Program exit
                                0C70  ;
                                0C70  ; COMPLETION CODES:
                                0C70  ;       NONE
                                0C70  ;
                                0C70  ; SIDE EFFECTS:
                                0C70  ;       NONE
                                0C70  ;
                                0C70  ;--
                                0C70
                                0C70  ERROR_EXIT:
                                0C70
                                0C70          $SETAST_S ENBFLG = #0         ; Disable AST's
        15 000A'CF    05    E0  0C79          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
                      7E    D4  0C7F          CLRL    -(SP)                   ; Set the time stamp flag
                 000F'CF    DF  0C81          PUSHAL  TEST_NAME               ; Set the test name
                      02    DD  0C85          PUSHL   #2                      ; Push the argument count
             00741039 8F    DD  0C87          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
          00000000'GF   04    FB  0C8D        CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                                0C94  10$:
       0192'CF    08    8E    C1  0C94        ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
                 014A'CF    D6  0C9A          INCL    ERROR_COUNT             ; Keep running error count
                      00    DD  0C9E          PUSHL   #0                      ; Push the time parameter
                 00A8'CF    DF  0CA0          PUSHAL  PROCESS_NAME            ; Push test name...
             000F0002 8F    DD  0CA4          PUSHL   #^XF0002                ; ...arg count...
             007410E2 8F    DD  0CAA          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
                 014A'CF    DD  0CB0          PUSHL   ERROR_COUNT             ; Finish off arg list...
                 00A8'CF    DF  0CB4          PUSHAL  PROCESS_NAME            ; ...
             00010002 8F    DD  0CB8          PUSHL   #^X10002                ; ...
             00748022 8F    DD  0CBE          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
    00000000'GF   0192'CF    FB  0CC4         CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                                0CCD
                                0CCD  ERROR_EXIT1:
                 014E'CF    D5  0CCD          TSTL    STATUS                  ; Did we exit with an error code?
                      09    12  0CD1          BNEQ    20$                     ; BR if we did
             007410E2 8F    D0  0CD3          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
```

```
                     014E'CF        0CD9  2036                    STATUS
                                    0CDC  2037 20$:
       014E'CF    10000000 8F   C8  0CDC  2038            BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                    0CE5  2039            $EXIT_S STATUS                  ; Exit in error
```

```
                        0CF0          .SBTTL  Exit Handler
                        0CF0  ;++
                        0CF0  ; FUNCTIONAL DESCRIPTION:
                        0CF0  ;       This routine handles cleanup at exit.  If the MODE logical name is
                        0CF0  ;       equated to "ONE", the routine will update the test flag in the
                        0CF0  ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                        0CF0  ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                        0CF0  ;       under test.
                        0CF0  ;
                        0CF0  ; CALLING SEQUENCE:
                        0CF0  ;       Invoked automatically by $EXIT System Service.
                        0CF0  ;
                        0CF0  ; INPUT PARAMETERS:
                        0CF0  ;       STATUS  contains the exit status.
                        0CF0  ;       FLAG    has synchronizing bits.
                        0CF0  ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                        0CF0  ;
                        0CF0  ; IMPLICIT INPUTS:
                        0CF0  ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                        0CF0  ;                 blocks for the device under test.
                        0CF0  ;
                        0CF0  ; OUTPUT PARAMETERS:
                        0CF0  ;       NONE
                        0CF0  ;
                        0CF0  ; IMPLICIT OUTPUTS:
                        0CF0  ;       Various files are de-accessed and the process name is reset.
                        0CF0  ;       If the MODE logical name is equated to "ONE", the routine will update
                        0CF0  ;       the test flag in the UETINIDEV.DAT file depending on the
                        0CF0  ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                        0CF0  ;       block for each unit for the device under test.
                        0CF0  ;
                        0CF0  ; COMPLETION CODES:
                        0CF0  ;       NONE
                        0CF0  ;
                        0CF0  ; SIDE EFFECTS:
                        0CF0  ;       NONE
                        0CF0  ;
                        0CF0  ;--
                        0CF0
                        0CF0  EXIT_HANDLER:
                  0FFC  0CF0          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0CF2
                        0CF2          $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                        0CFB          $SETAST_S ENBFLG = #0           ; Disable AST's
                        0D04          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                        0D04                  RSLLEN = BUFFER_PTR,-
                        0D04                  RSLBUF = FAO_BUF
      001C'CF    20     8A  0D1D          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
 001C'CF    4F 8F       91  0D22          CMPB    #^A70/,BUFFER           ; Is this a one shot?
              03        13  0D28          BEQL    10$                     ; BR if yes...
            00B8        31  0D2A          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                            0D2D  10$:
 03 000A'CF      02     E0  0D2D          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
            00AF        31  0D33          BRW     END_UPDATE              ; Else forget it
                            0D36  20$:
    5A    16A0'CF       DE  0D36          MOVAL   INI_RAB,R10             ; Set the RAB address
     1E AA       02     90  0D3B          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
```

```
      10 AA   16E4'CF    06     28   0D3F           MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                     0D46           $GET    RAB = (R10)               ; Go back to the DDB record
                      75 50     E9   0D4F           BLBC    R0,UPDATE_FAILED          ; If failure then forget it
                   1E AA  00     90   0D52           MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
  5B    0198'CF   00000198'8F     C1   0D56           ADDL3   #UNIT_LIST,UNIT_LIST,R11  ; Set the unit block list header
                         59     D4   0D60           CLRL    R9                        ; Init a counter
                                     0D62   UNIT_LOOP:
                         01     E1   0D62           BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
                   02 0B AB          0D64                   UETUNT$B_FLAGS(R11),10$
                         59     D6   0D67           INCL    R9                        ; Count testable units
                                     0D69   10$:
                      5B 6B     C0   0D69           ADDL2   (R11),R11                 ; Next unit block
          00000198'8F    5B     D1   0D6C           CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                         ED     12   0D73           BNEQ    UNIT_LOOP                 ; BR if not
                         59     D5   0D75           TSTL    R9                        ; Any testable units?
                         12     12   0D77           BNEQ    20$                       ; BR if yes...
           0020'CF    4E 8F     90   0D79           MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                     0D7F           $UPDATE RAB = (R10)               ; ...here
                      3C 50     E9   0D88           BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                     0D8B   20$:
                      5B 6B     C0   0D8B           ADDL2   (R11),R11                 ; Next unit block
          00000198'8F    5B     D1   0D8E           CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                         4E     13   0D95           BEQL    END_UPDATE                ; BR if yes
                                     0D97           $GET    RAB = (R10)               ; Get a record
                      24 50     E9   0DA0           BLBC    R0,UPDATE_FAILED          ; If error then forget it
              001C'CF    20     8A   0DA3           BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
           001C'CF    55 8F     91   0DA8           CMPB    #^A7U/,BUFFER             ; Is it a UCB record?
                         35     12   0DAE           BNEQ    END_UPDATE                ; BR if not
                         01     E0   0DB0           BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable...
                   D6 0B AB          0DB2                   UETUNT$B_FLAGS(R11),20$
           0020'CF    4E 8F     90   0DB5           MOVB    #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                     0DBB           $UPDATE RAB = (R10)               ; ...here
                      C4 50     E8   0DC4           BLBS    R0,20$                    ; Look at the next record if no error
                                     0DC7   UPDATE_FAILED:
                      0C AA     DD   0DC7           PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                         50     DD   0DCA           PUSHL   R0                        ; ...to tell of the failure
                    03D1'CF     DF   0DCC           PUSHAL  INIDEV_UPDERR
                         01     DD   0DD0           PUSHL   #1
                         00     EF   0DD2           EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
             7E    50    03          0DD4                   #STS$S_SEVERITY,R0,-(SP)
         6E    00741130 8F     C8   0DD7           BISL2   #UETP$_TEXT,(SP)          ; ...to our message
         C0000000'GF     05     FB   0DDE           CALLS   #5,G^LIB$SIGNAL
                                     0DE5   END_UPDATE:
                         00     DD   0DE5           PUSHL   #0                        ; Set the time flag
                    000F'CF     DF   0DE7           PUSHAL  TEST_NAME                 ; Push the test name
                         02     DD   0DEB           PUSHL   #2                        ; Push arg count
                         00     EF   0DED           EXTZV   #STS$V_SEVERITY,-         ; Push the proper exit severity...
                         03          0DEF                   #STS$S_SEVERITY,-
             7E    014E'CF          0DF0                   STATUS,-(SP)
         6E    00741080 8F     C8   0DF4           BISL2   #UETP$_ENDEDD,(SP)        ; ...and use it in our message code
                         04     DD   0DFB           PUSHL   #4
                      51 5E     D0   0DFD           MOVL    SP,R1
                                     0E00           $PUTMSG_S MSGVEC = (R1)           ; Output the message
                                     0E0F           $SETPRN_S PRCNAM = ACNT_NAME      ; Reset the process name
                                04   0E1A           RET                               ; That's all folks!
                                     0E1B
                                     0E1B           .END    UETDR7800
```

```
                                                   K  1
UETDR7800                    VAX/VMS UETP  EVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03  VAX/VMS Macro V04-00        Page  53
Symbol table                                                           5-SEP-1984 04:35:16  [UETPSY.SRC]UETDR7800.MAR;1      (19)

$$.TAB                  = 00001820 R    03      DEV_NAME                  000000BF R    03
$$.TABEND               = 00001864 R    03      DIAG_CNTRL_MESS           00001591 R    03
$$.TMP                  = 00000000              DIAG_READ_INT             0000060E R    02
$$.TMP1                 = 00000001              DIAG_REA_PKT              00001390 R    03
$$.TMP2                 = 0000006A              DIAG_WRIT_INT             0000061E R    02
$$.TMPX                 = 00000016 R    04      DIAG_WRI_PKT              00001370 R    03
$$.TMPX1                = 0000000D              DIAG_WRT_CNTRL            00000639 R    02
$$T1                    = 00000000              DIAG_WRT_PKT              00001570 R    03
$$T2                    = 00000006              DIB                       000000CE R    03
ACC$L_FINALSTS          = 00000004              DIB$B_DEVCLASS          = 00000004
ACNT_NAME                 00000000 R    02      DIB$B_DEVTYPE           = 00000005
ALL_SET                   000003E6 R    05      DIB$K_LENGTH            = 00000074
ARGS                      000001B0 R    03      DIB$W_UNIT              = 0000000C
ARG_COUNT                 00000192 R    03      DIBBUF                    000000D6 R    03
BADQUE                    00000472 R    02      DR750                     000003CD R    02
BADQUEUE                  00000A2C R    05      DR780                     000003C9 R    02
BADRPKT                   00000200 R    03      DRIOSTAT                  00000227 R    03
BEGIN_MSGM              = 00000020              DT$_DR750                 ********   X  05
BEGIN_MSGV              = 00000005              DUMMY_FAB                 0000173C R    03
BUFBLK                    0000022F R    03      DUMMY_RAB                 0000178C R    03
BUFBLKSIZ               = 00001080              DVI$_DEVNAM             = 00000020
BUFFER                    0000001C R    03      EFN1                    = 00000001
BUFFER_PTR                00000014 R    03      EFN2                    = 00000004
BUFSIZ                  = 00000800              END_UPDATE                00000DE5 R    05
CCASTHAND                 00000C39 R    05      ERROR_COUNT               0000014A R    03
CC_FLGM                 = 00000001              ERROR_EXIT                00000C70 R    05
CC_FLGV                 = 00000000              ERROR_EXIT1               00000CCD R    05
CHAN                      00000004 R    03      ERR_FLGM                = 00000008
CHF$L_SIGARGLST         = 00000004              ERR_FLGV                = 00000003
CHF$L_SIG_ARG1          = 00000008              ESC                     = 0000001B
CHF$L_SIG_ARGS          = 00000000              EXIT_DESC                 00000182 R    03
CHF$L_SIG_NAME          = 00000004              EXIT_HANDLER              00000CF0 R    05
CLR_RAND_ENABLE           0000065C R    02      FAB$B_BID               = 00000000
CLR_SELF_PKT              00001350 R    03      FAB$B_FNS               = 00000034
CLR_SELF_TEST             000005F7 R    02      FAB$C_BID               = 00000003
CMDBLK                    000012D8 RG   03      FAB$C_BLN               = 00000050
CMDBLKDES                 000004DC R    02      FAB$C_SEQ               = 00000000
CMDBLKEND                 000015BC R    03      FAB$C_VAR               = 00000002
CMDBLKSIZ               = 000002E4              FAB$L_ALQ               = 00000010
CMDTBL                    000012AF R    03      FAB$L_DEV               = 00000040
CMDTBLSIZ               = 00000020              FAB$L_FNA               = 0000002C
CNTRLCMSG                 000002B4 R    02      FAB$L_FOP               = 00000004
COMMON                    00000C00 R    05      FAB$L_STS               = 00000008
CONTROLLER                00000031 R    02      FAB$L_STV               = 0000000C
CONT_DESC                 00000416 R    02      FAB$V_CHAN_MODE         = 00000002
CS                        0000013F R    02      FAB$V_CR                = 00000001
CS1                       000001A5 R    02      FAB$V_FILE_MODE         = 00000004
CS1L                    = 000000C4              FAB$V_GET                 00000001
CS2                       000001B7 R    02      FAB$V_LNM_MODE          = 00000000
CS3                       000001FC R    02      FAB$V_PUT               = 00000000
CS4                       0000020B R    02      FAB$V_UFO               = 00000011
DDB_RFA                   000016E4 R    03      FAB$V_UPD               = 00000003
DEAD_CTRLNAME             000002F5 R    02      FAB$V_UPI               = 00000006
DEV$V_TRM               = 00000002              FAB$W_GBC               = 00000048
DEVDEP_SIZE             = 00000000              FAO_BUF                   0000000C R    03
DEVDSC                    000000A0 R    03      FILE                      0000041E R    02
DEVNAM_LEN                0000016C R    03      FIND_IT                   000001E1 R    05
```

```
FLAG                           0000000A R     03        PID                             00000006 R     03
FOR$CNV_OUT_F                  ********   X   05        PKT1_AST                        00000878 R     05
FOUND_IT                       00000279 R     05        PKT_CHECK                       00000A46 R     05
FPAC_FLGM                    = 00000010                 PKT_CNT                         000001A8 R     03
FPAC_FLGV                    = 00000004                 PKT_COUNT                     = 0000000D
FREE                           00000675 R     02        PKT_TBL                         0000056C R     02
FREEQH                         000012E8 R     03        PMTSIZ                        = 00000019
FREE_PKT                       00001598 R     03        PROCESS                         00000041 R     02
GOBIT                          000012CF R     03        PROCESS_NAME                    000000A8 R     03
HALT                           0000066E R     02        PROCESS_NAME_FREE             = 00000008
HALT_PKT                       00001310 R     03        PROC_CONT_NAME                  0000008B R     05
HUNG_TEST                      000008E1 R     05        PROMPT                          00000459 R     02
ILLEGAL_REC                    00000362 R     02        QUAD_STATUS                     00000152 R     03
INADDRESS                      0000015A R     03        RAB$B_PSZ                     = 00000034
INIDEV_UPDERR                  000003D1 R     02        RAB$B_RAC                     = 0000001E
INI_FAB                        00001650 R     03        RAB$C_BID                     = 00000001
INI_RAB                        000016A0 R     03        RAB$C_BLN                     = 00000044
INPTQH                         000012D8 R     03        RAB$C_RFA                     = 00000002
INPUT1_BUF                     0000122F R     03        RAB$C_SEQ                     = 00000000
INPUT_BUF                      00000A2F R     03        RAB$L_CTX                     = 00000018
INPUT_ITMLST                   00000195 R     02        RAB$L_FAB                     = 0000003C
IO$M_CTRLCAST                  ********   X   05        RAB$L_PBF                     = 00000030
IO$_READVBLK                   ********   X   05        RAB$L_ROP                     = 00000004
IO$_SETMODE                    ********   X   05        RAB$L_STS                     = 00000008
IO$_STARTDATA                  ********   X   05        RAB$L_STV                     = 0000000C
IOC$GW_XFMXRATE                ********   X   05        RAB$V_PMT                     = 0000001E
IO_COMPLETE                    0000092A R     05        RAB$W_RFA                     = 00000010
ITERATION                      00000176 R     03        RAB$W_RSZ                     = 00000022
LC_BITM                      = 00000020                 RANBUF                          00000A12 R     05
LIB$SIGNAL                     ********   X   05        RANDOM1                         0000016E R     03
MAX_DEV_DESIG                = 0000000A                 RANDOM2                         00000172 R     03
MAX_PROC_NAME                = 0000000F                 RATE_BUF                        00000208 R     03
MAX_UNIT_DESIG               = 00000005                 RATE_DESC                       0000021F R     03
MBCHAN                         00000002 R     03        RATE_FLOAT                      00000217 R     03
MODE                           00000133 R     02        READ                            000005A0 R     02
MSG_BLOCK                      0000017E R     03        READ_CHAIN                      000005A7 R     02
NAME_LEN                     = 0000000F                 READ_CHA_PKT                    000014A0 R     03
NAME_TBL                       000004E4 R     02        READ_DDI                        0000062E R     02
NEW_NODE                       000001A0 R     03        READ_DDI_PKT                    000013B0 R     03
NOOP                           00000607 R     02        READ_PKT                        000014D0 R     03
NOOP_PKT                       000012F0 R     03        READ_SIZE                     = 00000800
NOUNIT_SELECTED                0000033C R     02        RECORD                          0000042A R     02
NO_CTRLNAME                    000002D5 R     02        REC_SIZE                      = 00000028
NO_OF_POS_PKTS               = 00000011                 RESERVED                        000005DC R     02
NO_RMS_AST_TABLE               00000170 R     02        RESTART                         00000667 R     05
NRAT_LENGTH                  = 00000014                 RMS$_BLN                        ********   X   02
ONEMIN                         000003FE R     02        RMS$_BUSY                       ********   X   02
OTS$CVT_TI_L                   ********   X   05        RMS$_CDA                        ********   X   02
OUTADDRESS                     00000162 R     03        RMS$_EOF                        ********   X   05
OUTPUT_BUF                     0000022F R     03        RMS$_FAB                        ********   X   02
PACK_REMOVED                   000001AC R     03        RMS$_FACILITY                 = 00000001
PAGES                        = 00000009                 RMS$_FNF                        ********   X   05
PASS                           0000017A R     03        RMS$_RAB                        ********   X   02
PASS_MSG                       00000396 R     02        RMS_ERROR                       00000BCA R     05
PC1                            00000A4C R     05        RMS_ERR_STRING                  00000438 R     02
PC1...                       = 0000056C R     02        SAFE_TO_UPDM                  = 00000004
PC2...                       = 0000067C R     02        SAFE_TO_UPDV                  = 00000002
```

```
SEC$M_EXPREG                   ********    X    05         SYS$QIO                        ********  GX    05
SEC$M_GBL                      ********    X    05         SYS$QIOW                       ********  GX    05
SET_RAND_ENABLE                0000064A R       02         SYS$SETAST                     ********  GX    05
SET_SELF_PKT                   00001330 R       03         SYS$SETEF                      ********  GX    05
SET_SELF_TEST                  000005E7 R       02         SYS$SETIMR                     ********  GX    05
SHR$_ABENDD                  = 000010E0                    SYS$SETPRN                     ********  GX    05
SHR$_BEGIND                  = 00001038                    SYS$SETSFM                     ********  GX    05
SHR$_ENDEDD                  = 00001080                    SYS$TRNLOG                     ********  GX    05
SHR$_OPENIN                  = 00001098                    SYS$UPDATE                     ********  GX    05
SHR$_TEXT                    = 00001130                    SYS$WAITFR                     ********  GX    05
SS$_BADPARAM                 = 00000014                    SYS$WAKE                       ********  GX    05
SS$_CONTROLC                 = 00000651                    SYSIN_FAB                      000015BC R      03
SS$_NORMAL                   = 00000001                    SYSIN_RAB                      0000160C R      03
SS$_NOSUCHSEC                = 00000978                    TENSEC                         00000406 R      02
SS$_SSFAIL                   = 0000045C                    TERMQH                         000012E0 R      03
SS$_TIMEOUT                  = 0000022C                    TEST_DATA                      0000022F R      03
SS$_WASSET                   = 00000009                    TEST_END                       00000982 R      05
SSERROR                        00000AE7 R       05         TEST_HUNG                      000004A0 R      02
SS_SYNCH_EFN                 = 00000003                    TEST_NAME                      0000000F R      02
START_DATA_FAILED              0000028F R       02         TEST_OVERM                   = 00000002
STATUS                         0000014E R       03         TEST_OVERV                   = 00000001
STR$UPCASE                     ********    X    05         TEXT_BUFFER                  = 00000084
STS$K_ERROR                  = 00000002                    THREEMIN                       000003F6 R      02
STS$K_INFO                   = 00000003                    TIME_IT                        00000654 R      05
STS$K_SUCCESS                = 00000001                    TTCHAN                         00000000 R      03
STS$K_WARNING                = 00000000                    UETDR7800                      00000000 RG     05
STS$M_INHIB_MSG              = 10000000                    UETP                         = 00740000
STS$S_FAC_NO                 = 0000000C                    UETP$_ABENDD                 = 007410E0
STS$S_SEVERITY               = 00000003                    UETP$_ABORTC                 = 0074832B
STS$V_FAC_NO                 = 00000010                    UETP$_BEGIND                 = 00741038
STS$V_SEVERITY               = 00000000                    UETP$_COPY_LOG               = 007480B1
SUC_EXIT                       0000098A R       05         UETP$_COPY_LOG_ENDED         = 007480C1
SUPDEV_GBLSEC                  00000020 R       02         UETP$_COPY_LOG_LINE          = 007480B9
SUP_FAB                        000016EC R       03         UETP$_DATAER                 = 00748010
SYS$ASSIGN                     ********   GX    05         UETP$_DENOSU                 = 00748333
SYS$CANTIM                     ********   GX    05         UETP$_ENDEDD                 = 00741080
SYS$CLOSE                      ********   GX    05         UETP$_ERBOXPROC              = 00748020
SYS$CONNECT                    ********   GX    05         UETP$_FACILITY               = 00000074
SYS$CREMBX                     ********   GX    05         UETP$_OPENIN                 = 00741098
SYS$CREPRC                     ********   GX    05         UETP$_TEXT                   = 00741130
SYS$CRMPSC                     ********   GX    05         UETUNT$B_FLAGS               = 0000000B
SYS$DCLEXH                     ********   GX    05         UETUNT$B_TYPE                = 00000008
SYS$ERASE                      ********   GX    05         UETUNT$C_FAB                 = 00000110
SYS$EXIT                       ********   GX    05         UETUNT$C_INDSIZ              = 000001A4
SYS$EXPREG                     ********   GX    05         UETUNT$K_FAB                 = 00000110
SYS$FAO                        ********    X    05         UETUNT$K_RAB                 = 00000160
SYS$FAOL                       ********   GX    05         UETUNT$M_TESTABLE            = 00000002
SYS$GET                        ********   GX    05         UETUNT$T_FILSPC              = 00000014
SYS$GETCHN                     ********   GX    05         UETUNT$V_TESTABLE            = 00000001
SYS$GETDEV                     ********   GX    05         UETUNT$W_SIZE                = 00000009
SYS$GETDVI                     ********   GX    05         ULOAD_FAILED                   00000271 R      02
SYS$GETMSG                     ********   GX    05         UNIT_DESC                      0000040E R      02
SYS$INPUT                      00000184 R       02         UNIT_LIST                      00000198 R      03
SYS$LKWSET                     ********   GX    05         UNIT_LOOP                      00000D62 R      05
SYS$MGBLSC                     ********   GX    05         UNIT_NUMBER                    0000016A R      03
SYS$OPEN                       ********   GX    05         UNUSED_FUNC                  = 00006030
SYS$PUTMSG                     ********   GX    05         UPDATE_FAILED                  00000DC7 R      05
```

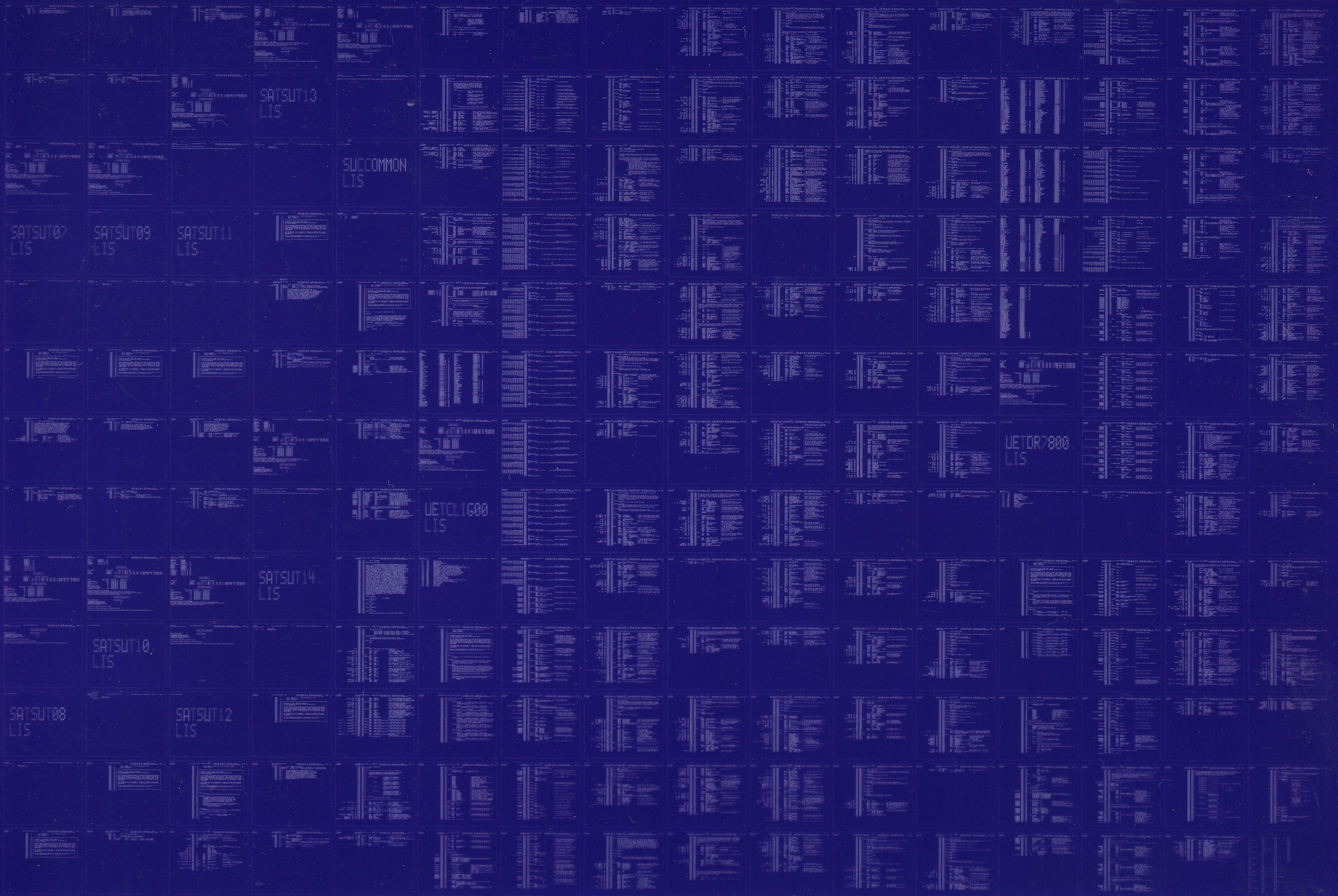
```
WRITE                            000005B4 R      02
WRITE_CHAIN                      000005BC R      02
WRITE_CH_PKT                     000013D0 R      03
WRITE_DEV_CNTRL                  000005CA R      02
WRITE_PKT                        00001400 R      03
WRITE_SIZE                     = 00000800
X                              = 00000080
XF$B_CMT_RATE                  = 00000018
XF$B_PKT_CMDCTL                = 0000000A
XF$K_CMT_LENGTH                = 00000020
XF$K_PKT_CBDMBC                = 00000003
XF$K_PKT_CLRTST                = 00000007
XF$K_PKT_DIAGRD                = 0000000B
XF$K_PKT_DIAGRI                = 00000009
XF$K_PKT_DIAGWC                = 0000000C
XF$K_PKT_DIAGWI                = 0000000A
XF$K_PKT_HALT                  = 0000000F
XF$K_PKT_NOINT                 = 00000002
XF$K_PKT_NOP                   = 00000008
XF$K_PKT_RD                    = 00000000
XF$K_PKT_RDCHN                 = 00000001
XF$K_PKT_SETTST                = 00000006
XF$K_PKT_UNCOND                = 00000000
XF$K_PKT_WRT                   = 00000002
XF$K_PKT_WRTCHN                = 00000003
XF$L_PKT_DSL                   = 0000001C
XF$M_CMT_DIPEAB                = 00000002
XF$M_CMT_SETRTE                = 00000001
XF$M_PKT_FREQPK                = 00000008
XF$S_PKT_FUNC                  = 00000004
XF$V_PKT_CISEL                 = 00000003
XF$V_PKT_FUNC                  = 00000000
XF$V_PKT_INTCTL                = 00000006
XFLDR_COPY_FINISH                00000123 R      02
XFLDR_COPY_LINE                  0000010F        02
XFLDR_COPY_START                 000000F3 R      02
XFLDR_HUNG                       00000077 R      02
XFLDR_LOG                        000000AF R      02
XFLDR_SYS$ERROR                  00000060 R      02
XFLDR_SYS$ERROR_DESC             0000006F R      02
XFLDR_SYS$ERROR_FAB              000017D0 R      03
XFLDR_SYS$ERROR_LENGTH         = 0000000F
XFLDR_SYS$ERROR_RAB              00001820 R      03

                                         +----------------+
                                         ! Psect synopsis !
                                         +----------------+

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000000  (     0.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
RODATA                           0000067C  (  1660.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                           00001864  (  6244.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                          00000023  (    35.)  04 (  4.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DR78                             00000E1B  (  3611.)  05 (  5.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
```

```
                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                    Page faults   CPU Time       Elapsed Time
-----                    -----------   --------       ------------
Initialization                   37    00:00:00.07    00:00:00.87
Command processing              138    00:00:00.73    00:00:03.63
Pass 1                          606    00:00:27.66    00:00:56.53
Symbol table sort                 0    00:00:02.43    00:00:04.94
Pass 2                          551    00:00:07.45    00:00:12.63
Symbol table output              46    00:00:00.34    00:00:00.44
Psect synopsis output             2    00:00:00.03    00:00:00.03
Cross-reference output            0    00:00:00.00    00:00:00.00
Assembler run totals           1382    00:00:38.72    00:01:19.08

The working set limit was 2000 pages.
165248 bytes (323 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1637 non-local and 79 local symbols.
2154 source lines were read in Pass 1, producing 45 object records in Pass 2.
71 pages of virtual memory were used to define 63 macros.

                                       +--------------------------+
                                       ! Macro library statistics !
                                       +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SHRLIB]UETP.MLB;1                   2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                   0
_$255$DUA28:[SYSLIB]STARLET.MLB;2               56
TOTALS (all libraries)                          58

1959 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETDR7800/OBJ=OBJ$:UETDR7800 MSRC$:UETDR7800/UPDATE=(ENH$:UETDR7800)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

68

0426 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


SATSUT13
LIS

SUCCOMMON
LIS

SATSUT07
LIS

SATSUT09
LIS

SATSUT11
LIS

UETDR7800
LIS

UETCLIG00
LIS

SATSUT14
LIS

SATSUT10
LIS

SATSUT08
LIS

SATSUT12
LIS

0427 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY